(19)日本国特許庁（ＪＰ）　(12) **公開特許公報（Ａ）**　(11)特許出願公開番号

特開平10－78881

(43)公開日　平成10年(1998)３月24日

(51)Int.Cl.⁶		識別記号	庁内整理番号	ＦＩ		技術表示箇所
Ｇ０６Ｆ	9/46	３６０		Ｇ０６Ｆ	9/46	３６０Ｂ
						３６０Ｆ
	13/00	３５７			13/00	３５７Ｚ

審査請求　未請求　請求項の数30　ＯＬ　（全 28 頁）

(21)出願番号　特願平9－170755

(22)出願日　平成９年(1997)６月26日

(31)優先権主張番号　０８／６７０６８４

(32)優先日　1996年６月26日

(33)優先権主張国　米国（US）

(71)出願人　595034134
サン・マイクロシステムズ・インコーポレイテッド
Ｓｕｎ　Ｍｉｃｒｏｓｙｓｔｅｍｓ，　Ｉｎｃ．
アメリカ合衆国　カリフォルニア州　94303　パロ　アルト　サン　アントニオ　ロード　901

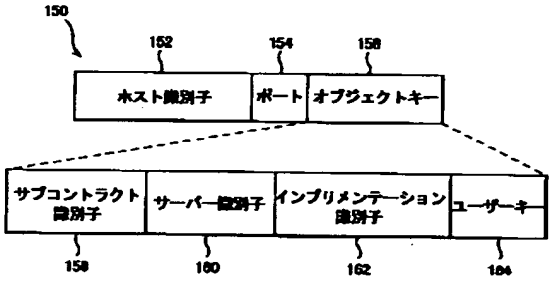
(72)発明者　スウィー　プーン　リム
アメリカ合衆国，　カリフォルニア州，　マウンテン　ヴュー，　スティアリン　ロード　405，　アパートメント　50

(74)代理人　弁理士　長谷川　芳樹　（外４名）

最終頁に続く

(54)【発明の名称】　オブジェクト呼出の性能を改善するためのメソッドと装置

(57)【要約】
【課題】分散クライアント／サーバー・ベースのコンピューティングシステムにおいて同一プロセスおよび異なるプロセス呼出用の異なる呼出経路を利用することによってコンピューティングオーバーヘッドを削減するデータ構造、メソッド、および装置を開示する。
【解決手段】発明の一側面では、リクエストするクライアントと同一プロセスを共有しないサーバントへのコールはトランスポート層を介して伝送され、またリクエストするクライアントと同一プロセスを共有するサーバントへのコールは当該サーバントに直接にパスされることによってトランスポート層をバイパスする。発明の別の側面では、リクエストの知的な伝送を容易にするために、異なるリモートおよびローカルメソッドテーブルが提供される。



1

【特許請求の範囲】

【請求項1】トランスポート層と、該トランスポート層の上にあるクライアント側の、複数のローカルメソッドテーブルと複数のリモートメソッドテーブルとを含むメソッドテーブルディスパッチ層、とを含む、クライアントオブジェクトからサーバントオブジェクトへコールをディスパッチするためのディスパッチメカニズムを有する分散クライアント／サーバー・ベースのコンピューティングシステムにおいて、第1セットの選択されたクライアント表示がローカルメソッドテーブルと関連すると共に第2セットの選択されたクライアント表示がリモートメソッドテーブルと関連し、クライアントからサーバントへコールを伝送するメソッドが、

該クライアントとサーバントが同一プロセスを共有しないときは、リモートメソッドテーブルと該トランスポート層を使って該コールを伝送するステップ、および

該クライアントとサーバントが同一プロセスを共有するときは、ローカルメソッドテーブルを使い、かつ該トランスポート層をバイパスすることによって該コールを伝送するステップ、を含むメソッド。

【請求項2】関連オブジェクトを独特なメソッドで識別するオブジェクトレファレンスを利用するように構成された分散クライアント／サーバー・ベースのコンピューティングシステムにおいて、

サービスのためのリクエストを示す複数のクライアント表示であって、各オブジェクトレファレンスは関連するクライアント表示を有し、選択されたクライアント表示は複数の異なるオブジェクトレファレンスと関連し得るように成したクライアント表示、

呼び出しリクエストがトランスポート層を介して伝送されるように構成された、第1セットの該クライアント表示に関連したリモートディスパッチメソッド、を識別するように構成されたリモートメソッドテーブル、および呼び出しリクエストがトランスポート層を介して伝送されることなくサーバントにパスするように構成された、第2セットの該クライアント表示に関連したローカルディスパッチメソッド、を識別するように構成されたローカルメソッドテーブル、を含む構成。

【請求項3】該オブジェクトレファレンスがそれぞれ関連クライアント表示を識別するように構成された第1ポインタと、該メソッドテーブルの中の関連する一つを識別するように構成された第2ポインタとを含むように成した、請求項2に記載の構成。

【請求項4】各メソッドテーブルは関連スタブを識別するように構成された複数のポインタを含み、かつ該メソッドテーブルの中の選択された一つへのポインタを含む各オブジェクトレファレンスが、該選択されたメソッドテーブルによって指示された関連スタブを有するように成した請求項2または3に記載の構成。

【請求項5】複数のリモートメソッドテーブルが、該第1セットのクライアント表示に関連するリモートディスパッチメソッドを識別するように構成されると共に、複数のローカルメソッドテーブルが、該第2セットのクライアント表示に関連するローカルディスパッチメソッドを識別するように構成された請求項2、3または4に記載の構成。

【請求項6】該複数のクライアントの少なくとも一部が、関連リモートメソッドテーブルと関連ローカルメソッドテーブルの両者を有するように成した請求項2～5のいずれかに記載の構成。

【請求項7】クライアント表示に関連するリモートメソッドテーブルとローカルメソッドテーブルの少なくとも一方を有する複数のクライアント表示を持つ分散クライアント／サーバー・ベースのコンピューティングシステムにおいて、

サーバントオブジェクトを識別するオブジェクトレファレンスをナロー化するための、クライアントプロセス内で受け取られるリクエスト、を受け取るステップ、

該リクエストによって識別される該サーバントオブジェクトが該クライアントプロセスにあるか否かを決定するプロセス、および該リクエストによって識別された該サーバントオブジェクトが該クライアントプロセスにある時は、ローカルメソッドテーブルを識別するナロー化されたオブジェクトレファレンスを作成するステップ、を含む、オブジェクトレファレンスをナロー化するメソッド。

【請求項8】該リクエストによって識別された該サーバントオブジェクトが該クライアントプロセスにないと決定されたときに、リモートメソッドテーブルを識別するナロー化されたオブジェクトレファレンスを作成するように成した請求項7に記載のメソッド。

【請求項9】クライアント表示に関連するローカルmテーブルとリモートmテーブルの一方を有する複数のクライアント表示を持つ分散クライアント／サーバー・ベースのコンピューティングシステムにおいて、

サーバントオブジェクトを識別するオブジェクトレファレンスをアンマーシャルするための、クライアントプロセス内で受け取られるアンマーシャル用リクエストを受け取るステップ、

該リクエストによって識別された該サーバントオブジェクトが該クライアントプロセスにあるか否かを決定するステップ、

該リクエストによって識別された該サーバントオブジェクトが該クライアントプロセスにあると決定されたときは、ローカルメソッドテーブルを識別するアンマーシャルされたオブジェクトレファレンスを作成するステップ、および該リクエストによって識別された該サーバントオブジェクトが該クライアントプロセスにないと決定されたときは、リモートメソッドテーブルを識別するアンマーシャルされたオブジェクトレファレンスを作成す

るステップ、を含む、オブジェクトレファレンスをアンマーシャルするメソッド。

【請求項10】オブジェクトレファレンスをアンマーシャルするための該リクエストが、クライアント表示に関連する該ローカルメソッドテーブルと該リモートメソッドテーブルの一方を使ってクライアント表示のアンマーシャル関数をコールするステップを備えるように成した請求項9に記載のメソッド。

【請求項11】該クライアント表示がそれに関連する複数の異なるローカルメソッドテーブルを有する、クレーム9または10に記載のメソッドであって、更に、該クライアント表示に関連するローカルメソッドテーブルの中の特定の一つを選択するステップを含み、また該アンマーシャル関数への該コールが該選択されたローカルメソッドテーブルによって行なわれるように成した、メソッド。

【請求項12】クライアント表示に関連するローカルスタブとリモートスタブの一方を有する複数のクライアント表示を持つ分散クライアント／サーバー・ベースのコンピューティングシステムにおいて、

該ストリンジファイされた(stringified) オブジェクトレファレンスを、マーシャルバッファによるデコードに適した二進ストリングに変換するステップ、

該二進ストリングをカプセル化するためにマーシャルバッファを取得するステップ、および該マーシャルバッファの内容をアンマーシャルするメソッドを呼び出すステップであって、該アンマーシャルは請求項9、10または11のいずれか一つのメソッドを利用することによって達成されるように成した呼出ステップ、を含む、ストリンジファイされたオブジェクトレファレンスをデストリンジファイするメソッド。

【請求項13】該ストリンジファイされたオブジェクトはＡＳＣＩＩストリンジファイドオブジェクトである請求項12に記載のメソッド。

【請求項14】分散クライアント／サーバー・ベースのコンピューティングシステムにおいて、

その関数が該クライアント／サーバー・ベースのオブジェクト指向オペレーティングシステムによって実行されるべき追加コードを含むか否かを示すように構成されたローカルフックを含む、該クライアントに関連するクライアント表示、によって指示された関数をコールするステップ、

該クライアント表示によって指示された該関数が、該クライアントに対してローカルプロセスにあるスタブ関数であるローカルスタブ関数か否かを決定するステップ、および該クライアント表示によって指示された該関数が該ローカルスタブ関数であると決定されたときは、該ローカルスタブ関数を実行するステップ、を含む、サーバントをクライアントから呼び出すメソッド。

【請求項15】該ローカルスタブ関数を実行するステップは、更に、下記のステップを備えるように成した、請求項14に記載のメソッド：追加コードを実行すべきことを該クライアント表示の該ローカルフックが示すか否かを決定するステップ、

該クライアント表示の該ローカルフックがアクティブであると決定されたときに、該クライアント表示に関する情報を収集するように構成された、該クライアント表示のプレローカルディスパッチメソッド、をコールするステップ、

該クライアント表示から該サーバントを取得するステップ、

該サーバントにおいて識別された、ローカルディスパッチメソッドであるメソッド、をコールするステップ、および該プレローカルディスパッチメソッドに関連した、該クライアント表示のポストローカルディスパッチメソッド、をコールするステップ。

【請求項16】該プレローカルディスパッチメソッドと該ポストローカルディスパッチメソッドの間で情報をパスするように構成された総称的ホルダーが該プレローカルディスパッチメソッドの該コールにパスされるように成した請求項15に記載のメソッド。

【請求項17】該プレローカルディスパッチメソッドが該クライアントからコールを受け取るための該サーバントを準備するように成した請求項15または16に記載のメソッド。

【請求項18】該ポストローカルディスパッチメソッドが該サーバントを非活性化するように成した請求項15、16、または17に記載のメソッド。

【請求項19】分散オブジェクトコンピューティングシステム内の分散サーバーオブジェクト上で定義されたオブジェクトメソッドを呼び出すためにコンピュータ読取り可能コードが具体化されているコンピュータ使用可能媒体を備えたコンピュータプログラムプロダクトであって、該分散オブジェクトコンピューティングシステムは関連オブジェクトを独特のメソッドで識別するオブジェクトレファレンスを利用するように構成され、該コンピュータプログラムプロダクトは、

サービスのためのリクエストを示す複数のクライアント表示であって、各オブジェクトレファレンスは関連するクライアント表示を有し、選択されたクライアント表示が複数の異なるオブジェクトレファレンスと関連し得るようなクライアント表示、

第1セットの該クライアント表示に関連する、呼出リクエストがトランスポート層を介して伝送されるように構成されたリモートディスパッチメソッド、を識別するように構成されたリモートメソッドテーブル、および第2セットの該クライアント表示に関連する、呼出リクエストがトランスポート層を介して伝送されることなくサーバントにパスするように構成されたローカルディスパッチメソッド、を識別するように構成されたローカルメソ

5

ッドテーブル、を備えた、構成用のコンピュータ読取り可能プログラムコードを備えたコンピュータプログラムプロダクト。

【請求項20】該オブジェクトレファレンスが、それぞれ、関連するクライアント表示を識別するように構成された第1ポインタと、該メソッドテーブルの中の関連する一つを識別するように構成された第2ポインタとを含むように成した、請求項19に記載の、構成用のコンピュータ読取り可能プログラムコードを備えたコンピュータプログラムプロダクト。

【請求項21】各メソッドテーブルが、関連スタブを識別するように構成された複数のポインタを含み、該メソッドテーブルの中の選択された一つへのポインタを含む各オブジェクトレファレンスが、該選択されたメソッドテーブルによって指示される関連スタブを有するように成した請求項19または20に記載の、構成用のコンピュータ読取り可能プログラムコードを備えたコンピュータプログラムプロダクト。

【請求項22】該複数のクライアント表示の少なくとも一部が、関連リモートメソッドテーブルと関連ローカルメソッドテーブルの両者を有するように成した請求項19、20、または21のいずれかに記載の、構成用のコンピュータ読取り可能プログラムコードを備えたコンピュータプログラムプロダクト。

【請求項23】分散オブジェクトコンピューティングシステム内の分散サーバントオブジェクト上で定義されたオブジェクトメソッドを呼び出すためにコンピュータ読取り可能コードが具体化されているコンピュータ使用可能媒体を備えたコンピュータプログラムプロダクトであって、該分散オブジェクトコンピューティングシステムは、クライアント表示に関連するリモートメソッドテーブルとローカルメソッドテーブルの少なくとも一方を有する複数のクライアント表示を持ち、該コンピュータプログラムプロダクトは該コンピュータシステム内に下記ステップをもたらすためのコンピュータ読取り可能プログラムコードを備えたプロダクト：該サーバントオブジェクトを識別するオブジェクトレファレンスを処理するためのリクエストを受け取るステップであって、該ナロー化リクエストはクライアントプロセス内で受け取られ、処理用の該リクエストは、該オブジェクトレファレンスをナロー化するためのリクエストと、該オブジェクトレファレンスをアンマーシャルするためのリクエストの一方であるステップ、

該リクエストによって識別された該サーバントオブジェクトがクライアントプロセスにあるか否かを決定するステップ、

該リクエストによって識別された該サーバントオブジェクトが該クライアントプロセスにあると決定されたときは、ローカルメソッドテーブルを識別する処理されたオブジェクトレファレンスを作成するステップ、および該

6

リクエストによって識別された該サーバントオブジェクトが該クライアントプロセスにないと決定されたときは、リモートメソッドテーブルを識別する処理されたオブジェクトレファレンスを作成するステップ。

【請求項24】オブジェクトレファレンスをナロー化するための該リクエストが、それに関連する該ローカルメソッドテーブルと該リモートメソッドテーブルの一方によってクライアント表示のナロー関数をコールするステップを備えた、請求項23に記載の、コンピュータ読取り可能プログラムコードを備えたコンピュータプログラムプロダクト。

【請求項25】該クライアント表示が、それに関連する複数の異なるローカルメソッドテーブルを有する、請求項23または24に記載の、コンピュータ読取り可能プログラムコードを備えたコンピュータプログラムプロダクトであって、該メソッドは、更に、該クライアント表示に関連する該ローカルメソッドテーブルの中の特定の一つを選択するステップを備え、かつ該デュプリケート関数へのコールは、該選択されたローカルメソッドテーブルによって行なわれるように成した、コンピュータプログラムプロダクト。

【請求項26】オブジェクトレファレンスをアンマーシャルするための該リクエストが、関連する該ローカルメソッドテーブルと該リモートメソッドテーブルの一方によってクライアント表示のアンマーシャル関数をコールするステップを備えた、請求項23に記載の、コンピュータ読取り可能プログラムコードを備えたコンピュータプログラムプロダクト。

【請求項27】該クライアント表示がそれに関連する複数の異なるローカルメソッドテーブルを有する、請求項23または26に記載の、コンピュータ読取り可能プログラムコードを備えたコンピュータプログラムプロダクトであって、該メソッドは、更に、該クライアント表示に関連する該ローカルメソッドテーブルの中の特定の一つを選択するステップを備え、かつ該アンマーシャル関数への該コールが該選択されたローカルメソッドテーブルによって行なわれるように成した、コンピュータプログラムプロダクト。

【請求項28】分散オブジェクトコンピューティングシステム内の分散サーバーオブジェクト上で定義されたオブジェクトメソッドを呼び出すためにコンピュータ読取り可能コードが具体化されているコンピュータ使用可能媒体を備えたコンピュータプログラムプロダクトであって、該コンピュータシステム内に下記のステップをもたらすためのコンピュータ読取り可能プログラムコートを備えたコンピュータプログラムプロダクト：その関数が該クライアント／サーバー・ベースの分散オブジェクトコンピューティングシステムによって実行されるべき追加コードを含むか否かを示すように構成されたローカルフックを含む、該クライアントに関連するクライアント

表示、によって指示された関数をコールするステップ、該クライアント表示によって指示された関数が、該クライアントに対してローカルプロセスにあるスタブ関数であるローカルスタブ関数であるか否か、を決定するステップ、および該クライアント表示によって指示された該関数が該ローカルスタブ関数であると決定されたときに、該ローカルスタブ関数を実行するステップ。

【請求項29】該ローカルスタブ関数を実行するステップが、更に、下記のステップを備えるように成した、請求項28に記載の、コンピュータ読取り可能プログラムコードを備えたコンピュータプログラムプロダクト：追加コードを実行すべきことを該クライアント表示の該ローカルフックが示すか否か、を決定するステップ、

該クライアント表示の該ローカルフックがアクティブであると決定されたときに、該クライアント表示に関係する情報を収集するように構成された、該クライアント表示のプレローカルディスパッチメソッド、をコールするステップ、

該クライアント表示から該サーバントを取得するステップ、

ローカルディスパッチメソッドである、該サーバントの中で識別されたメソッド、をコールするステップ、および該プレローカルディスパッチメソッドに関連する、該クライアント表示のポストローカルディスパッチメソッド、をコールするステップ。

【請求項30】該プレローカルディスパッチメソッドと該ポストローカルディスパッチメソッドの間で情報をパスするように構成された総称的ホルダーが、該プレローカルディスパッチメソッドにパスされるように成した、請求項29に記載の、コンピュータ読取り可能プログラムコードを備えたコンピュータプログラムプロダクト。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、概して、分散オブジェクト指向コンピューティングシステムに関する。より詳細には、分散オブジェクトシステムにおけるオブジェクトの呼出の性能を改善するために構成されたメソッド、データ構造および装置を開示する。

【0002】

【従来の技術】ネットワークで連結された異なるコンピュータ上にオブジェクトが配置されたコンピューティング環境は、通常、クライアントーサーバーコンピューティング環境と呼ばれている。コンピュータの中には、他のコンピュータに対するサービスや機能の提供者として働くものがある。サービスや機能の提供者は「サーバー」として知られ、サービスや機能の消費者は「クライアント」と呼ばれている。クライアントーサーバーモデルは、同一コンピュータ上で動作する異なるプログラムが、或る保護されたメカニズムを介して互いに通信しており、かつサービスや機能の提供者および消費者として働いているケースに一般化することもできる。

【0003】

【発明が解決しようとする課題】そのような分散システムを提供する試みは、従来、クライアントーサーバーモデル（そこではサーバーオブジェクトが当該オブジェクトにリクエストするクライアントオブジェクトにインターフェイスを提供する）に基づくオブジェクト指向メソッド論を使って行なわれている。通常、そのような分散システムでは、サーバーは、データと関連メソッドから成るオブジェクトである。クライアントオブジェクトは、サーバーオブジェクトをコール（当該分散システムによって媒介される）することによって、そのサーバーオブジェクトの機能に対するアクセスを獲得する。サーバーオブジェクトがコールを受け取ると、それは適切なメソッドを実行して、結果をクライアントオブジェクトに送り返す。クライアントオブジェクトとサーバーオブジェクトは、各種分散オブジェクトを配置すると共にオブジェクト間の通信を確立するために使用されるオブジェクトリクエストブローカー（ＯＲＢ）を介して通信する。分散オブジェクトは、ネットワーク内の任意の場所、例えばクライアントのアドレス空間やクライアントマシン上の多重アドレス空間やネットワーク全体の多重マシンの中に存在することができる。

【0004】ソフトウェア産業は、これまで、オブジェクト管理グループ（ＯＭＧ）を形成することによって分散オブジェクトテクノロジーに対するニーズに対応してきた。OMGの目標は、オブジェクト管理アーキテクチャ（ＯＭＡ）を定義することで、それは4つの大きなコンポーネント、すなわちオブジェクトリクエストブローカー（ＯＲＢ）、オブジェクトサービス、共通ファシリティ、およびアプリケーションオブジェクトを持つ。オブジェクトリクエストブローカーは、基本的なオブジェクト通信と管理サービスとを提供し、それによって分散オブジェクトシステムの基礎を形成する。オブジェクトリクエストブローカーの規格は共通オブジェクトリクエストブローカーアーキテクチャ（ＣＯＲＢＡ）仕様の中に含まれる。

【0005】典型的なクライアントーサーバーシステム、つまり分散オブジェクトシステムでは、そのアーキテクチャは、クライアントとサーバーの相対位置に関係なく、所定の呼出経路を使用してクライアントからのリクエストをサーバーに伝達するように構成されている。すなわち、リクエストは、クライアントとサーバーが同一または異なるプロセスに存在するか否かにかかわらず、システムの実質的に同じ層を介して伝えられる。ローカルコール、すなわち同一プロセスに位置するサーバーへのコールと、リモートコール、すなわち異なるプロセスに位置するサーバーへのコールの両者に対するこのような同一経路を使用することは、非効率的である。特に、クライアントとサーバーが同一プロセスにあるとき

は、クライアントからサーバーへの転送のリクエストをマーシャルした後、そのリクエストにアンマーシャルするトランスポートレベル関数は必要がなく、コンピューティングオーバーヘッドの非効率な使用であることが分かる。従って、クライアントとサーバーが同一プロセスに位置するか異なるプロセスにあるかによって選ばれる異なる呼出経路を利用することによってコンピューティングオーバーヘッドを削減するメソッドが、オブジェクト呼出の総合的性能を改善するために望ましいであろう。

【0006】

【課題を解決するための手段】前記およびその他の目的を達成するために、また本発明の目的に従って、分散クライアント／サーバー・ベースのコンピューティングシステムにおいて異なる呼出経路を利用してコンピューティングオーバーヘッドを削減するためのメソッドと装置が開示される。発明の一側面では、リクエストするクライアントと同一プロセスを共有しないサーバントへのコールはトランスポート層を介して伝えられ、リクエストするクライアントと同一プロセスを共有するサーバントへのコールはサーバントに直接にパスされることによってトランスポート層をバイパスする。

【0007】発明の別の側面では、複数のクライアント表示、リモートメソッドテーブル、およびローカルメソッドテーブルを含む分散クライアント／サーバー・ベースのコンピューティングシステムは、独特なメソッドで関連オブジェクトを識別するオブジェクトレファレンスを利用するように構成される。各オブジェクトレファレンスは関連クライアント表示を有する一方、選択されたクライアント表示は複数の異なるオブジェクトレファレンスに関連することができる。リモートメソッドテーブルは、第1セットのクライアント表示に関連するリモートディスパッチメソッドを識別するように構成される。リモートディスパッチメソッドは、呼出リクエストがトランスポート層を介して伝えられるように構成される。対照的に、ローカルメソッドテーブルは、第2セットのクライアント表示に関連するローカルディスパッチメソッドを識別するように構成される。

【0008】本発明は、その更なる目的と利点と共に、添付の図面に関する下記説明を参照することによって最もよく理解されるであろう。

【0009】

【発明の実施の形態】本発明は分散オブジェクトシステムに関し、添付の図面に示すいくつかの好ましい実施例について説明する。本発明は、CORBAによって定義されたもの、または他の任意の適切な仕様を含め、任意の適切な分散オブジェクトシステムとの関連において実行することができる。しかしながら、説明の目的のため、本発明は、主として、1995年7月付のOMGからのCORBA仕様、改訂版2．0、に基づいて実装されたオブジェクトリクエストブローカー（ORB）の関連において記述される。図1は、本発明の実施に適した代表的な分散オブジェクトシステムの全体的アーキテクチャを概略説明する。図2は、或るクライアントからサーバントオブジェクトへのリクエストが、3つのレベルのディスパッチメカニズムを含むようなアーキテクチャ内で取り得るいくつかの可能な流れ経路を概略説明する。図3は、サーバントオブジェクトを参照するためにクライアントが使用できるオブジェクトレファレンスデータ構造の一つを示す。

【0010】分散オブジェクトシステム10は、図1に象徴的に示すように、通常、オブジェクトリクエストブローカー（ORB）11を含む。ORB11は、クライアントからのコールをサーバント（目標オブジェクト）に伝達してそのクライアントに応答を返すのに必要な場所と輸送メカニズムと設備のすべてを提供するが、これについては図2に関して以下に説明する。クライアントとサーバントは、同一プロセス内、同一マシン上の異なるプロセス内、または全く異なるマシン上に配置できる。この検討の目的のためには、クライアント20は、分散オブジェクト上のオペレーションを呼び出す任意のコードでよいので、分散オブジェクトやプロセスの形を取っても、取らなくてもよい。正常なオブジェクトインプリメンテーション14は、C++等の従来のオブジェクトプログラミング言語で定義されるオブジェクトタイプの表示である。様々な表示が可能である。例として、オブジェクトインプリメンテーション14は、アプリケーションディベロッパによって提供される単純なC++オブジェクトタイプでもよい。そのほか、オブジェクトタイプのためのインプリメンテーションは、視覚アプリケーションビルダー15内で開発することもできる。この視覚アプリケーションビルダーによって、開発者は既存のオブジェクトタイプをカタログからビジュアルに選択して、オブジェクトタイプ用の新しいインプリメンテーションを作成するために、あるオブジェクトによって提供されたサービスを別のオブジェクト（属性、引数等）によって必要とされるサービスに図式的に接続することが可能となる。

【0011】オブジェクト開発ファシリティ16は、分散オブジェクトの作成とインストレーションを簡略化するために使用できる。それを使って、ディベロッパオブジェクトを分散オブジェクトコードの中に「ラップ」するか、カプセル化する。このように、オブジェクト開発ファシリティ16は、ディベロッパオブジェクトをORBオブジェクトインプリメンテーション14に変換するために使用できる。この例ではORBオブジェクトインプリメンテーション14は、図中のその場所で示されるように、サーバーとして提示される。ディベロッパはインタフェース定義言語を使ってORBオブジェクト用インタフェースを定義し、そのオブジェクトの挙動を実装

するオブジェクトインプリメンテーションをディベロッパに提供した後、そのオブジェクトの開発ファシリティを使ってORBオブジェクトインプリメンテーション14を作成する。動作時には、このORBオブジェクトインプリメンテーション14を利用する、このORBオブジェクト（サーバントオブジェクト）のインスタンスが作成される。オブジェクト開発ファシリティも、或る時点でクライアントの役割を担うオブジェクトの作成のために使用できることを理解しなければならない。

【0012】クライアント20は、スタブ21、メソッドテーブルディスパッチ24、サブコントラクト層36、可能性としてフィルタ40、およびトランスポート層38を介してサーバントと通信する。スタブ21はサロゲート22、メソッドテーブル24、およおびスタブ関数25を含む。クライアント20は、最初、クライアントにはサーバーオブジェクトに見えるサロゲート22と通信する。そのほかに、クライアント20は、サロゲート22、メソッドテーブル24、およびスタブ関数25の代わりに、動的呼出インタフェース（DII）を介してサーバーオブジェクトと直接に通信してもよい。動的呼出インタフェース26を使用して、クライアント、例えばクライアント20は動的リクエストを構築できる。上記の層を利用してクライアントがサーバントにコールする一つの手順を、図2に関して以下により詳しく説明する。

【0013】サブコントラクト層36は、サブコントラクトを利用して特定のサブコントラクトによって指名(named) された各種サービス（または特徴やオブジェクトメカニズム）を実施するために、オブジェクトによって要求された機能を提供するが、これについては上記の1995年11月7日申請の米国特許出願第08／554，794号に更に詳しく記載されている。サブコントラクトは、個々のオブジェクトによって利用される分散オブジェクトシステムによって提供されるサービスの質を識別する。例えば、サブコントラクトは、特定のオブジェクトのために安全性の特徴が使用されることを識別するだろう。フィルタ40が使用される場合、それは圧縮、暗号化、トレース、またはデバッグ等の、オブジェクトに出入りする通信に適用すべき様々なタスクを実行するだろう。

【0014】トランスポート層38は、通常はクライアントと同一プロセスを共有しないサーバントに出入りする情報を、マーシャル、アンマーシャル、および物理的に輸送するために動作する。

【0015】標準インプリメンテーションスイート28（またはオブジェクトアダプタ）は、同一の方式でORBオブジェクト14と相互作用する例えばオブジェクトキーマネージメントとしてのサブコントラクトの集まりを表す。当然のことながら、一つのサブコントラクトは複数のインプリメンテーションスイートに属することができる。従って、他のインプリメンテーションスイートが異なるサブコントラクトを利用することが可能である。スケルトンは、静的スケルトン32や動的スケルトン30の形を取り得るが、これは、サーバントオブジェクト14によってリクエストされるフォーマットにリクエストを変換するために使用される。かくして、スケルトン32、30は適切なサーバントオブジェクト14をコールする。静的スケルトン32はインタフェース固有のオブジェクトインプリメンテーション14をコールするために使用され、動的スケルトン30は、一般に、インタフェース固有のオブジェクトが利用できないときに使用される。ORBインタフェース34は、すべてのORBに対して同一であってオブジェクトのインタフェースやオブジェクトアダプタに依存しない、直接にORBに頼るインタフェースである。ORBデーモン46は、オブジェクトサーバーがクライアントによって呼び出されたときにアクティブである旨保証する責任がある。オブジェクトサーバーを起動する技術は、米国特許出願第08／408，645号に開示されている。

【0016】安全プロトコル42は、インターネットインターORBプロトコルを保証して、トランスポート層38を介して安全なメソッドで情報を伝達するのに役立つ安全インタオペラビリティプロトコルである。これは一貫性保護、機密性等を意味するだろう。インターネットインターORBプロトコルは、通常、異なるマシン上のプロセス間で通信するプロトコルである。しかしながら、場合によって、インターネットインターORBプロトコルは同一マシン上のプロセス間で通信することができる。セキュリティサーバー54は、異なるコンピュータ上のプロセス間で使用されるサービスを保証するセキュリティアドミニストレーションサーバーである。

【0017】タイプコード／Anyモジュール44はタイプコードと「Any」オブジェクトを実装する。タイプコードはインタフェース定義言語（IDL）データタイプを記述し、クライアントとサーバー間のタイプ記述の伝達を可能にする。IDLデータタイプのインスタンスを「Any」オブジェクトによってカプセル化することができる。Anyオブジェクトは、カプセル化されたデータのタイプコードと、データの総称的コード化を参照する。

【0018】インプリメンテーションレポジトリ50は、オブジェクトサーバーに関する情報を格納するために使用される。具体的には、インプリメンテーションレポジトリ50は、サーバープロセスを開始するのに必要な情報を格納する。例えば、インプリメンテーションレポジトリ50は、サーバープログラムの場所、プログラムに対する任意の引数、およびプログラムに渡すべき任意の環境変数等の情報を格納する。

【0019】シンプルパーシステンス56は、追加コードの一部と共に、インタフェース定義言語（IDL）定

義タイプと、ＩＤＬコンパイラを介してそのＩＤＬタイプを実行する出力形とを使用するので、ＩＤＬ定義タイプをディスクから読み取ったりディスクへ書き込むことができる。ネームサーバー５２は、ＯＲＢオブジェクトを名前付け(name)するために使用される。或るクライアント、例えばクライアント２０は、ネームサーバー５２を使って希望のオブジェクトをネームで発見することができる。ネームサーバー５２はオブジェクトレファレンスを返し、次にそのレファレンスを使ってそのオブジェクトにリクエストを送ることができる。インタフェースレポジトリ４８（ＩＦＲ）は、分散オブジェクトシステム内のすべてのオブジェクトに対するすべてのインタフェースについて知識を持つ。

【００２０】メソッドテーブル（「ｍ－テーブル」）ディスパッチを使って、クライアントによって行なわれたリクエストは、図２に概略図示するように、途中、アーキテクチャの前記の様々な層を通ってサーバントに至る。リクエストはクライアントによって開始され、任意の形を取ることができる。リクエストの形は、クライアントを作成するために使用されるプログラミング言語の性質に大きく依存する。例として、クライアントがＣ＋＋言語で書かれている場合、リクエストはＣ＋＋メソッドコール６２の形を取ることができる。コールは、サロゲートの形を取る指定のオブジェクトレファレンスに対して行なわれる。サロゲートは、オブジェクトのインタフェースに適合するメソッドを含む。当業者にはよく理解されるように、分散オブジェクトシステム内の異なるロケーションで使用されるオブジェクトレファレンスは外観が大きく変わる場合がある。記載の実施例では、クライアント側オブジェクトレファレンスはデュアルポインタである（本明細書では「ファットポインタ(fat pointer」と呼ぶ）。ファットポインタは性質の異なる２つのポインタを含む。第１のポインタ、つまり第１の場所インジケータは、レファレンスドオブジェクトに関連するクライアント表示（「クライアントレア」）を指示する。第２のポインタ、つまり第２の場所インジケータは、レファレンスドオブジェクトに関連するメソッドテーブルディスパッチ２４のメソッドテーブルを指示する。ここで用いられる用語「ポインタ」はコンピュータネットワーク内の場所を識別するために使用されるばかりでなく、用語「ポインタ」は一般的な場所インジケータのことを指すためにも使用されることを理解しなければならない。クライアント表示は、呼出、ならびにＣＯＲＢＡで定義された「疑似」オブジェクトレファレンスオペレーションを支援するメソッドを持つオブジェクトである。これらのオペレーションは、これに限定されるものではないが、デュプリケートメソッド、リリーズメソッド、ナローメソッド、ハッシュメソッド、およびイズ－イクイバレントメソッドを含む。

【００２１】クライアントがコールを開始した後、そのコールはメソッドテーブルディスパッチメカニズム２４を使って処理される。かかる技術は米国特許出願第０８／３０７，９２９号に開示されている。メソッドテーブルディスパッチメカニズムは、スタブ関数２５に対するポインタ、つまり場所インジケータ、のリストを含むメソッドテーブルを使用するが、そのポインタの一つは呼び出すべきメソッドに関連している。スタブ関数２５はクライアントプロセスの「ネイティブ」言語による関数や手続きコールを受け取った後、サブコントラクト層３６かネイティブコールのいずれかを使って、結局、対応するサーバントオブジェクトをコールする。ネイティブ言語は、任意の適切な言語、例えばＣ＋＋のような言語でもよい。

【００２２】メソッドテーブルディスパッチ２４はそのメソッドコールを処理する適切なスタブ関数２５を決定した後、メソッドコールをその適切なスタブ関数２５と組合せる。メソッドコールを行なったクライアントがサーバントオブジェクトと同一プロセスにある場合、ローカルスタブ関数がコールされる。ローカルスタブ関数は、そのメソッドコールを直接にサーバントオブジェクト７８に送る。そのほかに、サーバントオブジェクトが異なるプロセス、すなわちリモートプロセスにある場合、リモートスタブ関数がコールされる。リモートスタブ関数はクライアント表示を呼び出し、クライアント表示がその呼出をサーバントオブジェクト７８に伝達する。

【００２３】サブコントラクト層３６によって実装されたサブコントラクトは、分散オブジェクトシステムで重要なオブジェクトの呼出と引数のパスの基本メカニズムを制御する論理モジュールである。サブコントラクト層３６によって実装されるサブコントラクトは、オブジェクトによって使用されるサービスの固有の質を決定する。サブコントラクトはサブコントラクト識別子によって独特なメソッドで識別され、その識別子は、通常、オブジェクトレファレンスに埋め込まれている。サービスの質とはサービス特性の集まりである。選択可能なサービス特性の中には、サーバーの作動、セキュリティ、トランザクション、フィルタ性、およびクリーンシャットダウンに関係する質がある。サブコントラクトは、一定の質のサービスが利用できるように構成されている。予め定められたサービスの質によって、個々のサービス特性の処理に伴うオーバーヘッドが削減される。実施に、「妥当な」または一般的に使用されるサービス特性の組合せがサブコントラクトによってサポートされる。しかしながら、所定の分散オブジェクトシステムの固有の要件を満たすためにサブコントラクトを作成してもよい。

【００２４】サブコントラクト層３６内の適切なサブコントラクトの識別は、そのサブコントラクトに独特な望ましい関数の識別と見做すことができる。例えば、マーシャル関数やアンマーシャル関数は、各サブコントラク

ト毎に定義される。サブコントラクトマーシャル関数は、別のアドレス空間や領域に伝送できるようにオブジェクトレファレンスをマーシャルするためにスタブによって使用される。オブジェクトレファレンスは、通常、トランスポート層38内のトランスポートメカニズムによって処理される。

【0025】T1、T2等のトランスポートメカニズムは、トランスポート層38の一部だが、サーバントオブジェクトに出入りする情報をマーシャルして物理的に輸送するために使用される。情報、すなわちオブジェクトレファレンスやリクエストは、所定の領域に適したプロトコルに転換される。例として、プロトコルは、それに限定されるわけではないが、イーサーネットプロトコルとインターネットインタオペラブルプロトコル（IIOPs）を含んでもよい。希なケースでは、プロトコルは、サーバー上で実装されるべき命令を伝送するために電子メールの使用を必要とすることもある。情報がマーシャルされた後、そのトランスポートメカニズムはオペレーティングシステム、デバイスドライバ、またはネットワーク（これらは、すべて、分散オブジェクトシステムのクライアント側によって使用されるハードウェア70の一部）の任意の組合せを介して情報を輸送する。トランスポートメカニズムでは、情報を所定領域に適したプロトコルに転換する必要があるが、中には異なる領域に対する情報のコード化を必要としないトランスポートメカニズムもある。情報の発生場所以外の領域に適したプロトコルへの情報の転換を必要としないトランスポートメカニズムは「ドア」と呼ばれる。ドアは、基本的に、同一ホスト上の2つの異なるプロセス間のゲートウェイである。ドアを使用することによって、トランスポート層38内の標準インプリメンテーションに情報を転換するニーズが除去される。というのは、情報が同一ホストに残っているので領域を変更する必要がないために、異なるマシンによって使用され得るプロトコルに情報をコード化する必要がないからである。従って、情報を単に「平坦化(flattened out)」するか、あるいは異なるマシンによる使用のためにコード化されないストリームの中にマーシャルして、ホスト上の2つのプロセス間でパスすることができる。

【0026】一旦、情報が、クライアント側によって使用されるハードウェア70を介して輸送されると、その情報は、次に、分散オブジェクトシステムのサーバー側のハードウェア70に輸送される。一旦、情報がハードウェア70を通って流されると、分散オブジェクトシステムのサーバー側は、トランスポート層38の一部であるエンドポイントで情報を受け取るために、T1、T2等のトランスポートメカニズムを呼び出す。万一トランスポート層38によってエンドポイントが作成されない場合、トランスポート層38は、そのエンドポイントがサブコントラクト層36によって作成されるのに必要な機能を提供する。例として、ドアエンドポイントは、通常サブコントラクト層36によって作成され、一方、ネットワークとTCP/IPエンドポイントを含む他のエンドポイントは、通常、トランスポート層38によって作成される。エンドポイントの作成がサブコントラクト層36によるかトランスポート層38によるかにかかわらず、エンドポイントはトランスポート層38の中に「住む(live in)」、つまりその一部である。エンドポイントは、基本的に異なる領域から情報を受け取るポートである。トランスポート層38内のエンドポイントが、異なる領域から輸送された情報を受け取った後、そのエンドポイントは、次に、その情報をトランスポート層38からサブコントラクト層36にディスパッチする。サブコントラクト層36、より具体的にはその情報を受け取るサブコントラクト層36内のサブコントラクトが、次にその情報をスケルトンとサーバントにディスパッチする。

【0027】サブコントラクト層36は、それが受け取った情報の少なくとも一部をアンマーシャルするための機能を提供する。すなわち、サブコントラクト層36はリクエストの少なくとも一部をアンマーシャルする。次に、リクエストがスケルトン31にディスパッチされ、スケルトンがそのリクエストを、サーバントオブジェクト78によって要求されるインプリメンテーションの固有のフォーマットに変換する。スケルトンは、上記のように、静的スケルトンでも動的スケルトンでもよい。

【0028】一般に、リモートリクエストのルートは、上記のように、クライアント側とサーバー側を通らなければならない。メソッドコール62が受け取られると、メソッドテーブルディスパッチ層24を使って、リクエストをマーシャルしてそれを別の領域に輸送するために準備するトランスポート層38内のトランスポートメカニズムを選択する前に、適切なサブコントラクトが識別される。ハードウェア70を介して、マーシャルされたリクエストがサーバー側に輸送され、そこでそのリクエストがトランスポート層38の一部のエンドポイント上で受け取られる。適切なエンドポイントがワイヤを伝って輸送された情報を受け取ると共に、情報がトランスポート層38からサブコントラクト層36にディスパッチされるので、サブコントラクト層はそれが受け取った情報を少なくとも部分的にアンマーシャルする機能を提供する。次に、サブコントラクトは、そのリクエストをスケルトン31にディスパッチし、スケルトンはそのリクエストを、サーバントオブジェクト78によって要求される固有のフォーマットに変換する。この経路を矢印77で示すが、これはリモートリクエストとローカルリクエストが共に取り得る経路である。

【0029】しかしながら、クライアントとサーバーがローカルプロセスにある場合、すなわちクライアントとサーバーが共に同一プロセスにある場合、上記の矢印7

7で示される経路の使用は必要以上に複雑である。クライアントとサーバーが同一プロセスにあることが分かっていれば、呼出し、つまりサービスのためのリクエストの流れ経路を短縮することができる。オブジェクトレファレンスを作成するときにローカルプロセスが識別された場合、クライアントであるものから同一ホスト上のサーバーにリクエストを送るために、短縮された流れ経路、つまり矢印75、76で示される経路を取ることができる。矢印76で示される経路を取る可能性の方が大きい。というのは、それが適切なサブコントラクトを識別するためにサブコントラクト層36を使用しているからである。しかしながら、適切なサブコントラクトをはっきり識別する必要のない状況では矢印75で示される経路を取ることができる。

【0030】ここで、図3を使って、オブジェクトレファレンスの一実施例を説明する。当業者にはよく知られているように、オブジェクトレファレンスは、それらが任意の時点で占めているプロセス内の場所によって様々な形を取ることができる。しかしながら、背景として、低いオーバーヘッドのオブジェクトアダプタを利用するシステムで使用するための代表的なオブジェクトレファレンスを図3に示す。ここで示されるインプリメンテーションでは、オブジェクトレファレンス150は、ホスト識別子152、ポート指定154、およびオブジェクトキー156を含む。オブジェクトキー156は、サブコントラクト識別子158、サーバー識別子160、インプリメンテーション識別子162、およびユーザーキー164を含む。ホスト識別子152は、ネットワーク内の特定のコンピュータを指し、ポート指定154は、通信用に使用すべき選ばれたコンピュータのポートを識別する。オブジェクトキー156は、望ましいサーバントオブジェクトをホストマシン上に配置するために使用される更なる識別用情報を提供する。

【0031】サーバー識別子160は、サーバントオブジェクトが常駐する特定のプロセスやプログラムを指示し、一方、ユーザーキー164は、サーバー識別子160によって指示されるプロセス内にサーバントを配置するために使用される唯一の数または文字列である。サブコントラクト識別子158は、特定サブコントラクトのプロトコルと、サーバントに対するその関連サービスとを生成するために使用され、インプリメンテーション識別子162は、そのサーバントオブジェクトと共に使用すべきインタフェースのインプリメンテーションを指示する。

【0032】次に図4を参照して、mテーブルディスパッチメカニズムを更に詳しく説明する。上述のように、記載の実施例では、呼び出すべきサーバントを識別するためにクライアントによって使用されるオブジェクトレファレンスは「デュアル」ポインタまたは「ファットポインタ」90と見做すことができる。すなわち、2つの「正常な」ポインタを含むポインタ構造である。ファットポインタ90はCORBAオブジェクトレファレンスと見做すこともできる。各ポインタのサイズはシステムの要件によって支配される。ファットポインタ90の第1ポインタは、クライアント表示（「クライアントレア」）94を指示するクライアント表示ポインタ90aである。ファットポインタ90の第2ポインタは、クライアント表示94に関連するmテーブル24を指示するmテーブルポインタ90bである。上記の説明のように、クライアント表示94は、オブジェクト呼出とCORBAで定義された「疑似」オブジェクトレファレンスオペレーション（デュプリケートおよびナローメソッドを含む）とをサポートするメソッドを持つオブジェクトである。クライアント表示94はクライアント上のサーバントオブジェクトの表示と見做される。かくして、クライアント表示94はサーバントと関連している。各オブジェクトレファレンスは単一の関連クライアント表示94を持つが、各クライアント表示94は一つ以上の異なるオブジェクトレファレンスに関連することができる。クライアント表示94のサブコントラクトは、クライアント表示94が一つ以上のオブジェクトレファレンスと関連するか否かを決定する。

【0033】各クライアント表示94は、一般に、関連するmテーブルを持つ。関連mテーブルはローカルmテーブルでも、リモートmテーブルでもよい。或る実施例では、多重のローカルまたはリモートmテーブルが利用される。クライアント表示の中には一つ以上の関連mテーブルを持つものがある。クライアント表示94がローカルmテーブルとリモートmテーブルの両者を格納、維持するか否かは、クライアント表示94が関連するサブコントラクトのインプリメンテーションに依存する。一般に、mテーブル24は、ディスパッチの決定のために使用されるメソッドへのポインタのアレイを含む。クライアント表示94はスタブ、つまりスタブ関数25にパスされる引数であるから、ディスパッチはクライアント表示94から独立している。スタブ関数25は、先に記載したように、クライアントプロセスのネイティブ言語による関数コールを受け取った後、サポートライブラリを使って、結局、対応するサーバントオブジェクトをコールする。ディスパッチを決定するために使用されるmテーブルは希望のオペレーション用のスタブ関数25を指示することができる。別法として、スタブ関数25を直接指示するのではなく、mテーブルは、ディスパッチを決定するために使用されるスタブ関数25へのポインタのアレイを含む他のmテーブルを指示してもよい。

【0034】分散オブジェクトシステムの中の各インタフェースは少なくとも一つのmテーブルに関連することができる。しかしながら、インスタンスの中には関連するmテーブルを持たないものもあることを理解しなければならない。mテーブルは、通常、コンパイルされた、

つまり静的なスタブと共に使用される。従って、コンパイルされたスタブが存在しない実施例では、インタフェースは関連mテーブルを持たないかもしれない。通常、各インタフェースは単一のローカルmテーブルと単一のリモートmテーブルに関連している。もっとも、或る場合には、一つのインタフェースが一つ以上のローカルmテーブルに関連することができる。

【0035】一般に、mテーブルは多くの異なる表示を持つことができる。例として、或る実施例では、mテーブルは「フラット」、つまりmテーブルは直接にスタブ関数を指示できる。他の実施例では、mテーブルはツリー構造、すなわちmテーブルは、直接にmテーブルによって指示されないスタブ関数25を指示し得るデータ構造を指示するだろう。所定のインタフェースの各メソッドに対するポインタは、そのインタフェースに関連するmテーブルからアクセス可能であることを理解しなければならない。従ってmテーブルがツリー構造の場合、メソッドに関連するポインタを発見するためにmテーブルのツリー構造をたどる必要があるかもしれない。

【0036】下記に更に詳しく説明するように、提案のアーキテクチャの一つの利点は、ローカルとリモートの手続きの間の区別を可能にすることによって、クライアントによって受け取られるリクエストの最適な流れ経路や呼出経路の決定が可能になることである。例として、クライアントとサーバーが同一プロセスに存在しない場合、受け取られたリクエストは少なくともクライアントとサーバーの両者のトランスポート層を介して伝えられ、サーバープロセスが異なるホスト上にあるときは、ハードウェア層も通ることになる。しかしながら、クライアントとサーバーが互いにローカルにある場合、すなわち、同一プロセスを共有するときは、受け取られたリクエストをトランスポート層とハードウェア層とを介して送る必要はない。クライアントとサーバーが同一プロセスにある場合は、修正された、より短い流れ経路を利用できるので、受け取られたリクエストはクライアント側のサブコントラクト層とmテーブルディスパッチ層のどちらからも適切なサーバントオブジェクトに伝達することができる。不必要なときに、受け取ったリクエストをトランスポート層とハードウェア層とを介して送ること（リクエストと回答のマーシャルとアンマーシャルが必要）は、効率、従ってオブジェクトの呼出性能を損なうことになる。サーバントがクライアントと同一プロセスにあるか否かを効率的に識別できるようにオブジェクトレファレンスを構成することによって、クライアントとサーバーが同一プロセスに存在する場合でも、短い流れ経路の利用が可能となる。従って、オブジェクト呼出の性能が改善される。

【0037】次ぎに図5を参照して、mテーブルディスパッチを使用したオブジェクトの呼出に関するステップを説明する。最初、ファットポインタを使ってコールが行なわれる。記載の実施例では、C++コールの形を取る。もっとも、コールの性質は、勿論、通常はクライアントが書かれたプログラム言語の関数になる。ステップ100では、コールされたメソッドが、ファットポインタによって指示されたmテーブルに配置される。コールされたオブジェクトがローカルなメソッドの場合、ファットポインタによって指示されるmテーブルはローカルmテーブルになる。同様に、コールされたオブジェクトがリモートな場合、ファットポインタによって指示されるmテーブルはリモートmテーブルになる。ステップ105では、コールは、ステップ100で指示されたスタブ関数に対して行なわれる。記載の実施例ではC++コールであるコールは、ファットポインタによって識別されたクライアント表示と、コールパラメータとしてパスされた他の引数とを使って行なわれる。或る実施例では、追加の引数がパスされる。指示されたスタブ関数がコールされた後、プロセス制御は、指示されたスタブ関数がローカルスタブ関数にあるかリモートスタブ関数にあるかによって、異なる関数に分岐する。指示された関数がローカルスタブ関数にある場合、プロセス制御はステップ110に進み、そこでローカルスタブ関数が実行される。このローカル呼出は図4に関して以下により詳しく検討する。ローカルスタブがステップ110で実行された後、プロセスはステップ120の関数コールから戻る。ファットポインタによって指示された関数がリモートスタブ関数にある場合、プロセス制御は、ステップ105からステップ115に進み、そこでリモートスタブが実行される。リモートスタブがステップ115で実行された後、プロセス制御はその関数コールからの戻りであるステップ120に進む。

【0038】次に図6を参照して、ローカルスタブを使用して呼出を実施するための適切なメソッドを説明する。すなわち、図5のステップ110の一実施例を更に詳しく説明する。図2に関して、ローカルスタブを「実行」するプロセスは、経路75または76を介してサーバントを呼び出すプロセスと理解されるだろう。或るオペレーション、例えばオペレーションXのためのローカルスタブを実行するプロセスはステップ200で開始され、ローカルスタブによってパスされたクライアント表示のローカルフックがアクティブか否か決定される。フックがアクティブか否かの任意の適切な表示が使用できる。例として、ローカルフックはその一部であるクライアント表示がアクティブか否かを識別する2値のフラグでもよい。ローカルフックは、追加コード（例えばオブジェクトディベロッパによってカスタマイズされたコード）がクライアント－サーバーシステム内で実行できるメカニズムを提供する。すなわち、ローカルフックは、クライアント－サーバーシステムが追加コードを実行しなければならないか否かを示すようにセットできる。大抵の実施例では、この決定は実行時と対立するコンパイ

ル時に行なわれるので、ローカルフックが使用されないときは、フック設備の設置による性能の低下はないだろう。

【0039】ローカルフックがアクティブでないとき、それはローカルスタブを実行するプロセス内に含むべき追加コード、特にサブコントラクト固有のコードが存在しないことを示し、プロセス制御はステップ216に進み、そこで、インタフェース定義言語（IDL）で規定されたネームのエレメントを持つ新しいコンテクストオブジェクトが、オリジナルコールとスタブ関数へのコールからパスされたオリジナルコールリクエストから作成される。IDLは、オブジェクトのインタフェースを定義するために使われる言語である。コンテクストオブジェクトは、結合のリスト、例えば「USERNAME＝foo」および「HOSTNAME＝foo」の2つの文字列を含むリストである。更に一般的にはコンテクストオブジェクトは、優先情報や分散オブジェクトシステムの構成に関する情報を格納するために使用できる。コンテクストオブジェクトがステップ216で作成された後、プロセス制御はステップ218に進み、そこで、サーバントがクライアント表示から直接に取得される。ローカルスタブは使用中なので、クライアントとサーバントが同一プロセスを共有することが分かる。従って、サーバントは、当業者には理解されるように、標準技術を使って容易に取得できる。記載のコンテクストオブジェクト作成ステップ216はオプションのステップで、希望なら除去してもよい。通常、コンテクスト作成ステップ216は、関連メソッドがIDLによるコンテクストクローズで規定されないときは削除される。関連メソッドがコンテクストクローズで規定されない場合、コンテクスト引数はオリジナルコールまたはスタブ関数へのコールの一部としてパスされないだろう。コンテクストオブジェクト作成ステップ216が省略された場合、プロセス制御はステップ200（そこでクライアント表示のローカルフックがアクティブでない旨決定された）から直接にステップ218に進み、そこで、サーバントがクライアント表示から直接取得される。

【0040】サーバントがステップ218で取得された後、ステップ218で識別されたサーバー用のオペレーションXに対応するメソッドがステップ220でコールされる。ステップ220でのサーバーコールに対する引数は、通常、クライアント表示を除いて、スタブ呼出ステップ105でローカルスタブにパスされるものと同一引数である。新しいコンテクストがオプションのステップ216で作成された場合、その新コンテクストもそのコールと共にオリジナルコンテクストの代わりにパスされる。ステップ220が完了した後、結果が返される。結果が返されたとき、新コンテクストは、それが作成された場合には、自動的に削除または破壊される。

【0041】クライアント表示のローカルフックがアクティブである旨ステップ200で決定された場合、それは追加コードを実行しなければならないことを示し、プロセス制御はステップ202に移り、そこで、クライアント表示のプレローカルディスパッチメソッドに対するコールが行なわれる。クライアント表示のローカルフックがアクティブなときは、プレローカルディスパッチメソッドが、ローカルディスパッチ、特にサーバントオブジェクトへのポインタによって要求されるクライアント表示に固有の情報を収集するために、ローカルディスパッチが起こる前にコールされる。プレローカルディスパッチは、サーバントポインタが確実に取得されることを確認するために使用される。更に、プレローカルディスパッチメソッドは、サーバントにコールを受け取るように準備させたり（活性化）、コーラーがサーバントを呼び出す許可を取るのを保証したり（セキュリティ）、トランザクションを格納したり、ロックを獲得したりするために、追加のサブコントラクト処理を実行することができる。正確な処理はクライアント表示に関連するサブコントラクトに依存することを理解しなければならない。プレローカルディスパッチメソッドに対するコールでは、ジェネリックホルダーがパスされる。総称的ホルダーは、プレローカルディスパッチメソッドからポストローカルディスパッチメソッドまでのローカルスタブの実行に関係する希望の情報をパスするために使用できる。総称的ホルダーは、サブコントラクトに関係するプレ呼出状態を保つための全体的記憶やスレッド固有の記憶の使用を避けるために用意される。総称的ホルダーは、そのホルダーにバリューを割り当てるプレローカルディスパッチメソッドにパスされる。一般的に、割り当てられる値はトランザクション識別子を含む構造へのポインタや、プレローカルディスパッチで割り当てられた記憶へのポインタ（通常、次のサーバントコール用ポストローカルディスパッチで割当て解除される）でもよい。

【0042】ステップ202におけるクライアント表示のプレローカルディスパッチメソッドに対するコールから、プロセス制御はステップ204に進み、そこで、プレローカルディスパッチコールが成功したか否か決定される。コールが不成功の場合、例外、つまりこの実施例ではC++例外が起こり、更なる処理は行なわれない。コールが成功した場合、プロセス制御はステップ206に進み、そこで、IDLで規定されたネームのエレメントを持つ新しいコンテクストオブジェクトが作成される。従って、プロセス制御はステップ208に進み、そこで、サーバントがクライアント表示から取得される。サーバントが取得された後、オペレーションXに対応するメソッドが、識別されたサーバントに対してステップ210でコールされる。すなわち、ステップ210ではローカルディスパッチが起こる。ステップ206～210は上述のステップ216～220のミラーであること

を理解しなければならない。従って、コンテクストオブジェクト作成ステップ216と同様に、コンテクスト作成ステップ206はオプションのステップである。

【0043】ローカルディスパッチがステップ210で実行された後、プロセス制御はステップ212に進み、そこで、ジェネリックホルダーがプレローカルディスパッチメソッドからパスされた状態で、クライアント表示のポストローカルディスパッチメソッドがコールされる。ポストローカルディスパッチメソッドはジェネリックホルダーに割り当てられた値（通常、ローカルコールに関連するデータを含む構造へのポインタを含む）を回復する。ポストローカルディスパッチメソッドは、事実上、プレローカルディスパッチ用のクロージャであり、プレローカルディスパッチオペレーションに対応するハウスキーピングオペレーションを実行させるためにコールされる。例として、ポストローカルディスパッチメソッドは、サーバントオブジェクトを非活性化したり、トランザクションを行なったり、ロックを解除することができる。一般に、プレローカルディスパッチとポストローカルディスパッチは、サブコントラクトのローカル呼出への参加や介入を可能にする。一旦、ローカルディスパッチプロセスを終了するためにポストローカルディスパッチメソッドがコールされると、ローカルスタブを実行するプロセスが完了する。

【0044】ローカルフックがコンパイル時に有効にされると、ステップ202～212、すなわちプレローカルとポストローカルディスパッチメソッドを利用するメソッドに対するコールが実行されることを理解しなければならない。ローカルフックがコンパイル時に有効にされないと、ステップ216～220、すなわちプレローカルとポストローカルディスパッチメソッドを利用しないメソッドへのコールが実行される。換言すれば、ステップ200の分岐、すなわちローカルフックがアクティブか否かの決定は、そのブランチがコンパイラによって予め決定できるので、必要がないかもしれない。従って、所定のブランチに関係するコードのみが生成される。

【0045】上述のように、知的なメソッドでオブジェクトレファレンスを構築することによって、クライアントとサーバーが同一プロセスにある場合に、より短い流れ経路を利用することが可能になる。知的なメソッドでオブジェクトレファレンスを構築することは、関係するオブジェクトレファレンス、すなわち新しいオブジェクトレファレンスの作成元であるオブジェクトレファレンスがサーバーに対してローカルであるか否かを決定することを意味する。新オブジェクトレファレンスの作成元であるオブジェクトレファレンスがサーバーに対してローカルである場合、ローカルmテーブルを指示するmテーブルポインタを持つファットポインタ、すなわちCORBAオブジェクトレファレンスが作成される。新オブジェクトレファレンスの作成元であるオブジェクトレファレンスがサーバーに対してローカルでない場合、リモートmテーブルを指示するmテーブルポインタを持つファットポインタが作成される。

【0046】適切なmテーブルを指示するかそれに添付されたmテーブルポインタを使ってファットポインタ、つまりCORBAオブジェクトレファレンスを作成するために、ファットポインタの作成時に、適切なmテーブルはローカルテーブルかリモートテーブルかに関する決定を行なわなければならない。オブジェクトレファレンスの作成のために使用される標準CORBA関数は、クリエートメソッド、デュプリケートメソッド、ナローメソッド、アンマーシャルメソッド、およびデストリンジファイ(destringify) メソッドを含む。すなわち、デュプリケートメソッド、ナローメソッド、アンマーシャルメソッド、およびデストリンジファイメソッドは、すべて、新オブジェクトレファレンスを構築するために使用できる。従って、これらの各メソッド（および新オブジェクトレファレンスを作成する他の任意のメソッド）において、サーバントが、作成されるオブジェクトレファレンスと同一プロセスにあるか否かに関する決定がなされるなら、より短い流れ経路でローカルリクエストを処理することが可能となるだろう。つまり、大抵の場合、かなりの割合のオブジェクト呼出がローカルになるので、短い流れ経路の使用は、先に検討したように、オブジェクト呼出の性能の改善に役立つ。

【0047】次に図7を参照して、本発明の一実施例による、サーバントがローカルかリモート火を決定するオブジェクトレファレンスを複製するプロセスを説明する。記載の実施例では、プロセスはステップ250で開始され、オブジェクトレファレンス、つまりオブジェクト「レフ」が関連するクライアント表示のデュプリケートメソッドに対するコールが行なわれる。複製されるオブジェクトレファレンスのmテーブルへのポインタは、クライアント表示のデュプリケートメソッドに対するコールの引数としてパスされる。すなわち、各クライアント表示は、mテーブルを引数と見做すデュプリケートメソッドを持つ。コールされるデュプリケートメソッドは、通常、オブジェクトレファレンスのクライアント表示によって識別される。デュプリケートメソッドがステップ250でコールされた後、プロセス制御はクライアント表示のレファレンスカンウントが増分されるステップ252か、クライアント表示のコピーが作成されるステップ256のいずれかに進むことができる。レファレンスカウントがステップ252で増分されるか、クライアント表示のコピーがステップ256で作成されるかは、クライアント表示のサブコントラクトが参照されたオブジェクトの可測性(accountability)をどのように管理するかによって決まる。当業者にはよく理解されるように、様々なメカニズムを使用して特定のクライアント

表示のために存在する顕著なオブジェクトレファレンスの数をカウントすることができる。例として、レファレンスカウンタを使って同一クライアント表示に関係するオブジェクトレファレンスの数が追跡される。すなわち、レファレンスカウントは、複製されたオブジェクトレファレンスがクライアント表示に関連するオブジェクトレファレンスの数を追跡するためにクライアント表示に対して作成される毎に増分される。別法として、オブジェクトレファレンスが作成される毎にクライアント表示の新しいコピーを作ることができる。オブジェクトレファレンスが作成される毎にクライアント表示のコピーを作成することによって、クライアント表示当たり一つしかオブジェクトレファレンスは存在しないので、オブジェクトレファレンスの数を容易にカウントすることができる。

【００４８】デュプリケートメソッドのための可測性メカニズムがレファレンスカウントの増分化を含む場合、プロセス制御はステップ２５０のデュプリケートメソッドに対するコールからステップ２５２のレファレンスカウントの増分化に移る。レファレンスカウントの増分化によって、ＯＲＢは新、すなわち複製されたオブジェクトレファレンスが作成されようといることを知らされる。ステップ２５４では、クライアント表示へのポインタと、デュプリケートメソッドへのコールに対する引数として受け取られたｍテーブルポインタの両者を含む新オブジェクトレファレンスが作成される。サブコントラクトは、レファレンスカウントがステップ２５２で増分されたことによって新オブジェクトレファレンスが作成されたことを知っている。複製用オブジェクトレファレンスがステップ２５４で作成された後、プロセス制御は進み、新しく作成されたオブジェクトレファレンスを返す。レファレンスカウントを増分するステップと、オリジナルのクライアント表示を含むオブジェクトレファレンスを作成するステップとは、オブジェクトレファレンスを複製するいくつかのメソッドに特有であることを理解しなければならない。

【００４９】デュプリケートメソッドのための可測性メカニズムが、クライアント表示のコピーの作成を含む場合、プロセス制御は、ステップ２５０のクライアント表示のデュプリケートメソッドへのコールからステップ２５６のクライアント表示のコピーの作成に移る。クライアント表示のコピーが作成された後、プロセス制御はステップ２５８に進み、そこで、クライアント表示のコピーへのポインタと、デュプリケートメソッドへのコールから受け取られたｍテーブルポインタとを含む新しいオブジェクトレファレンスが作成される。新オブジェクトレファレンスがステップ２５８で作成された後、プロセス制御は前進して新しく作成されたオブジェクトレファレンスを返す。クライアント表示のコピーを作成するステップとクライアント表示のコピーへのポインタを含むオブジェクトレファレンスを作成するステップとは、オブジェクトレファレンスを複製する場合に使用されるいくつかのメソッドに特有であることを理解しなければならない。

【００５０】各サブコントラクトは異なるデュプリケートメソッドをそのサブコントラクトに関連するクライアント表示に結びつけることを理解しなければならない。例として、上述のように、レファレンスカウントを増分させるステップと、オリジナルのクライアント表示を含むオブジェクトレファレンスを作成するステップとは、上記メソッドの一つの一部であり、クライアント表示のコピーを作成するステップと、クライアント表示のコピーを含むオブジェクトレファレンスを作成するステップとは、別の上記メソッドの一部である。これらのメソッドは、新しいオブジェクトレファレンスが作成されたときは常に、サーバントオブジェクトに接触するサブコントラクトに関係する追加処理を含むことができる。

【００５１】デュプリケートメソッドは、通常、クライアント表示に関連する他のオペレーション、またはメソッド、によってコールされる。例として、デュプリケートメソッドは、ナローメソッド、アンマーシャルメソッド、およびデストリンジファイメソッド等（すべてクライアント表示に属するメソッド）によってコールできる。先に検討したように、これらのメソッドがコールされるときに、サーバントがクライアントに対してローカルプロセスにあるか否かを決定することによって、作成されたオブジェクトレファレンスをローカルまたはリモートｍテーブルのいずれかと関連させることが可能になる。それによってローカルプロセスの識別、従ってリクエストを処理するためのより短い流れ経路の利用が可能になるので、オブジェクト呼出の性能が改善される。

【００５２】次に図８を参照して、本発明の一実施例による、サーバントがローカルかリモートかを決定するオブジェクトレファレンスをナロー化する(narrowing) メソッドを説明する。オブジェクトレファレンスをナロー化することは、基本的に、オブジェクトレファレンスを一般タイプから特定、つまり目標タイプに変換することである。オブジェクトレファレンスをナロー化することは、オブジェクトレファレンスのコピー作成のために使用される一メソッドである。異なるサブコントラクトは異なるナローメソッドを持つ異なるクライアント表示を持ち得ることを理解しなければならない。オブジェクトレファレンスをナロー化するプロセスはステップ２８０で開始され、そこで、クライアント表示のナローメソッドが、関連ローカルまたはリモートｍテーブルを目標タイプの引数として使用してコールされる。ステップ２８２で、ナロー化が達成されるか否かについて決定が行なわれる。ナロー化が達成できないと決定された場合、プロセス制御はステップ２８３で失敗状態を返す。しかし、ナロー化が達成できる旨ステップ２８２で決定され

た場合、プロセス制御はステップ２８６に進み、サーバントがローカルプロセスにあるか否かの決定が行なわれる。サーバントがローカルプロセスにあるか否かの決定は、サーバントがローカルプロセスにあるか否かを決定する目的で特別に書かれた関数、例えば上記のＮＥＯ分散オブジェクトシステムのクライアント表示のイズ_ローカルメソッドをコールすることによって行なわれる。クライアント表示のイズ_ローカルメソッドは、各サブコントラクトとクライアント表示に特有である。ローカルプロセスが単にローカルマシン上で実行されるプロセスを指すだけではないことを理解しなければならない。そうではなく、サーバントがローカルプロセスにあるか否かの決定は、サーバントが同一プロセスにあるか否かの決定なのである。ステップ２８６の結果が肯定的な場合、すなわちサーバントがローカルプロセスにある場合、クライアント表示のデュプリケートメソッドが、ローカルmテーブルを持つナロー化されたオブジェクトレファレンスを作成するために、ローカルmテーブルを引数として使用してステップ２８８でコールされる。ナローメソッドにパスされたmテーブルは目標インタフェース、つまりナロー化オブジェクトレファレンスが適合すべきインタフェースに関係するmテーブルであることを理解しなければならない。クライアント表示のデュプリケートメソッドは、指定のmテーブルポインタを含むオブジェクトレファレンスを作成し、図7に関して先に記載されたように、作成されたオブジェクトレファレンスを返すだろう。次に、新オブジェクトレファレンスはナローオペレーションの結果として返される。

【００５３】サーバントがローカルプロセスにないとの決定がステップ２８６で行われた場合、プロセス制御はステップ２９０に進み、そこで、リモートmテーブルを引数として使用して、クライアント表示のデュプリケートメソッドに対するコールが行われる。これが、リモートmテーブルポインタを持つナロー化されたオブジェクトレファレンスを作成する。再度述べておくが、デュプリケートメソッドが、図5に関して先に説明したオブジェクトレファレンスを作成するのである。新オブジェクトレファレンスは、次に、ナローオペレーションの結果として返される。

【００５４】クライアント表示が一つのリモートmテーブルと多数のローカルmテーブルに関連する場合、一旦、サーバントがローカルプロセスにあるとステップ２８６で決定されると、プロセス制御は、ローカルmテーブルを持つナロー化されたオブジェクトレファレンスを作成するステップのステップ２８８と同じ一般的な目的に役立つステップ２８８ａに進む。しかしながら、ステップ２８８ａは、クライアント表示が関連する一つ以上のローカルmテーブルが存在するときは、ナロー化されたオブジェクトレファレンスの作成に使用すべき適切なローカルmテーブルの識別を含む。全体ステップ２８８ａはサーバントオブジェクトを実装するための適切なmテーブルを識別するステップであるステップ２９２を含む。適切なmテーブルの識別は、サブコントラクト、すなわちクライアント表示固有のステップである。例として、サブコントラクトに関連するインプリメンテーションレポジトリを使用して、クライアント表示によって表されたサーバントオブジェクトの実装に適したローカルmテーブルを配置することができる。多数の関連mテーブルを持つクライアント表示の場合、ナローメソッドへの引数としてパスされるローカルmテーブルは、そのローカルmテーブルがサーバントオブジェクトの実装に適したローカルmテーブルであると判明しない限り、通常使用されない。一旦、適切なローカルmテーブルが識別されると、クライアント表示のデュプリケートメソッドが、ローカルmテーブルポインタを持つナロー化オブジェクトレファレンスを作成するために、ローカルmテーブルを引数として使用してステップ２９２でコールされる。次に、ナロー化オブジェクトレファレンスがコーラーに返される。

【００５５】代表的な分散オブジェクト指向システムでオブジェクトレファレンスを作成する他のメカニズムも存在する。そのようなメカニズムの一つは、リクエストまたは回答の中で引用されるオブジェクトレファレンスのアンマーシャルである。当業者にはよく理解されるように、ネットワークの通信ラインまたはプロセス内通信ポートを介して転送される情報は、通常、マーシャルバッファで受け取られる。受信された情報は、選ばれたネットワークまたはプロセス内通信プロトコルに関連するフォーマットで着信する。アンマーシャルとは、マーシャルバッファで受け取った情報を、非ネットワーク環境で有意のフォーマットに変換するプロセスのことを指す。

【００５６】分散オブジェクト指向環境では、クライアントとサーバー間の大部分の通信は、リクエストや回答の目標であるオブジェクトの識別を意図するオブジェクトレファレンスで開始される。リクエストや回答も、通常、追加情報を含み、その情報は、目標オブジェクトへの伝達のためのインタフェースパラメータ、例外パラメータ、データ等の形を取ることができる。時には、リクエストや回答を伴うデータおよび／またはパラメータは、リクエストや回答を伝送するために使用されるのに反して、目標オブジェクトに伝達されると予想される追加のオブジェクトレファレンスを含むこともある。アンマーシャル用のメカニズムは、少なくとも一部分、目標オブジェクトレファレンスに基づいてリクエストを正しく伝送するとともに、目標オブジェクトによって使用できる形に追加情報を（必要に応じて）変換するように、構成しなければならない。かくして、オブジェクトレファレンスが目標オブジェクトに伝達すべきパラメータや他のデータの一部であるときは、アンマーシャル関数

は、目標プロセスに有意なオブジェクトレファレンスを効果的に作成しなければならない。これはオブジェクトレファレンスを作成する別の潜在的メカニズムであるから、「データ」オブジェクトレファレンスが、リクエストや回答を受け取っているプロセスに対する「ローカル」オブジェクトを指すのか「リモート」オブジェクトを指すかを決定することが大切である。

【００５７】次に図９を参照して、本発明の一実施例による、リクエストと回答に記載されるデータオブジェクトレファレンスをアンマーシャルするメソッドを説明する。データオブジェクトレファレンスのインタフェースタイプが分かっている限り、適切なタイプのデータオブジェクトレファレンス用のＯＲＢ提供のジェネリックアンマーシャル用ルーチンに対するコールがステップ３００で行なわれる。この例の目的では、データオブジェクトレファレンスに関連するインタフェースを「インタフェースＹ」と呼ぶ。アンマーシャルメソッドは、インタフェースＹ用マーシャルバッファと共に、インタフェースＹ用リモートｍテーブルへのポインタとインタフェースＹ用ローカルｍテーブルへのポインタの両者を持つＯＲＢ提供のジェネリックアンマーシャル用ルーチンをコールする。

【００５８】ＯＲＢによって提供されたジェネリックアンマーシャル用ルーチンがコールされた後、データオブジェクトレファレンスに関連するサブコントラクト識別子、すなわちサブコントラクトＩＤがステップ３０３で取得される。オブジェクトレファレンス内のサブコントラクト識別子の場所が分かっているので、「ピーク」メソッドを使用してデータオブジェクトレファレンス内からサブコントラクト識別子を取得できる。ピークメソッドは、抽出はしないが、マーシャルバッファの既知の場所からサブコントラクト識別子を読み取る。一旦、オブジェクトレファレンス用のサブコントラクトＩＤが分かると、オブジェクトレファレンスに対応するアンマーシャル関数が、サブコントラクトＩＤをキーとして使ってサブコントラクトレジストリで調査される。　ステップ３０６で求められたアンマーシャル関数は、インタフェースＹに関連する引数、すなわちリモートとローカルのｍテーブル、ならびにマーシャルバッファを使って、ステップ３０９でコールされる。アンマーシャル関数へのコールの後、プロセス制御はステップ３１２に進み、そこで、マーシャルバッファ内の情報からクライアント表示が作成される。次に、マーシャルバッファポインタがマーシャルバッファ内の次のアイテムに「加え」れらるか、移される。ステップ３１５では、作成されたクライアント表示のホスト識別子とサーバー識別子、すなわちホストＩＤとサーバーＩＤとが、現行プロセスのホストＩＤおよびサーバーＩＤと同一か否か決定される。それに従って、クライアント表示のローカルフラグが、次に、ステップ３１５でセットされる。すなわち、オブジェクトレファレンスから抽出されたホストＩＤおよびサーバーＩＤが現行プロセスのホストＩＤおよびサーバーＩＤとマッチした場合、ローカルフラグはローカルサーバントを示す状態にセットされる。さもなければ、ローカルフラグはリモートサーバントを示す状態にセットされる。ステップ３２０では、サーバントがローカルプロセスにあるか否か決定される。図８に関して先に説明したように、サーバントがローカルプロセスにあるか否かの決定は、サーバントがローカルプロセスにあるか否かを決定するために特別に書かれたクライアント表示固有関数をコールすることによって行なわれる。この場合、その関数がクライアント表示のローカルフラグをチェックする。サーバントがローカルプロセスにある場合、プロセス制御はステップ３２３に進み、クライアント表示のデュプリケートメソッドへのコールが、ローカルｍテーブルを持つアンマーシャルされたオブジェクトレファレンスを作成するために、ローカルｍテーブルを引数として使用して行なわれる。次に、新しく作成されたアンマーシャルされたオブジェクトレファレンスが返される。

【００５９】クライアント表示が、一つのリモートｍテーブルと多数のローカルｍテーブルに関連する場合、一旦、サーバントがローカルプロセスにある旨ステップ３２０で決定されると、プロセス制御はステップ３２３ａに進む。ステップ３２３ａは、ローカルｍテーブルを持つアンマーシャルされたオブジェクトレファレンスを作成するステップであるステップ３２３と同じ全体的な目的に役立つ。しかしながら、ステップ３２３ａは、クライアント表示が関連する一つ以上のローカルｍテーブルが存在するときは、ナロー化されたオブジェクトレファレンスの作成に使用するための適切なローカルｍテーブルの識別を含む。全体ステップ３２３ａは、サーバントオブジェクトの実装用の適切なｍテーブルを識別するステップのステップ３２４を含む。適切なｍテーブルの識別は、サブコントラクト、すなわちクライアント表示固有ステップである。複数の関連ｍテーブルを持つクライアント表示のケースでは、アンマーシャルメソッドへの引数としてパスされるローカルｍテーブルは、ローカルｍテーブルがサーバントオブジェクトの実装用の適切なローカルｍテーブルであると判明しない限り、通常、使用されない。一旦、適切なローカルｍテーブルが識別されると、クライアント表示のデュプリケートメソッドが、ローカルｍテーブルポインタを持つアンマーシャルされたオブジェクトレファレンスを作成するために、ローカルｍテーブルを引数として使用してステップ３２５でコールされる。次の、そのアンマーシャルされたオブジェクトレファレンスがコーラーに返される。

【００６０】サーバントがローカルプロセスにない旨ステップ３２０で決定された場合、プロセス制御はステップ３２０からステップ３２６に進み、そこで、リモート

mテーブルを持つアンマーシャルされたオブジェクトレファレンスを作成するために、クライアント表示のデュプリケートメソッドがリモートmテーブルを引数として使ってコールされる。ステップ326が完了すると、アンマーシャルされたオブジェクトレファレンスが返される。クライアント表示のデュプリケートメソッドに対するコールは図5に関して上記で説明されている。

【0061】次に図10を参照して、マーシャルバッファの情報からクライアント表示を作成する別のメソッドを、既知のタイプのインタフェース用のオブジェクトレファレンスをアンマーシャルする関連において説明する。この実施例では、オブジェクトレファレンスをアンマーシャルするプロセスは、クライアント表示を作成するステップまでは図9に記載のプロセスと同一である。クライアント表示の作成はステップ330で開始され、そこで、オブジェクトレファレンスに関係する情報がマーシャルバッファから抽出される。ステップ330は枠321aの一部であり、適切なマーシャル関数を調べるステップの、図7のステップ306から直接に続く。オブジェクトレファレンスに関係する情報がステップ330でマーシャルバッファから抽出された後、ステップ332で、参照されたサーバントオブジェクトレファレンス用のクライアント表示がすでの存在しているか否かの決定が行なわれる。決定の結果が否定的な場合、プロセス制御はステップ334に進み、そこで、クライアント表示が、ステップ330で先に抽出されたオブジェクトレファレンスに関係する情報を使って作成される。次に、ステップ315aで、新しく作成されたクライアント表示の中のホストIDとサーバーIDとが現行プロセスのホストIDおよびサーバーIDと同一か否かについての決定が行なわれる。一旦、決定が行なわれると、クライアント表示のローカルフラグがそれに従ってセットされる。サーバントが同一プロセスにあるか否かを決定するために使用されるメカニズムには、違いがあることを理解しなければならない。例として、ホスト内コールがサポートされない実施例では、ホストIDは使用されないだろう。一旦、ローカルフラグがセットされると、次にプロセス制御はステップ320に進む。このステップは、図9に関して上記に説明したように、適当なサーバントがローカルプロセスにあるか否かを決定するステップである。オブジェクトレファレンスのためのクライアント表示がすでに存在している旨ステップ332で決定された場合、プロセス制御はステップ332から直接にステップ320に進み、適切なサーバントがローカルプロセスにあるか否かの決定が行なわれる。一旦、プロセス制御が、適切なサーバントはローカルプロセスにあるか否かを決定するステップ、すなわちステップ320に到達すると、既知のタイプを持つインタフェース用のオブジェクトレファレンスをアンマーシャルするステップの残りは、図9に関して先に説明したものと同一ステップである。

【0062】図11を参照して、本発明の一実施例による、サーバントがローカルかリモートかを決定するオブジェクトレファレンスをデストリンジファイ(destringifying)するメソッドを説明する。オブジェクトレファレンスを「デストリンジファイ」するとは、ASCII文字列を受取用プロセスによって理解可能なオブジェクトレファレンスに変換することを意味する。すなわち、オブジェクトレファレンスのデストリンジファイは、ASCII文字列の「ディヘクシファイ(dehexifying)」を必要とする。デストリンジファイのプロセスは新オブジェクトレファレンスの作成をもたらし、記載の実施例ではステップ350で開始され、そこで、ストリンジファイされたオブジェクトレファレンスであるASCII文字列が、マーシャルバッファによってデコードするのに適した文字列に変換される。記載の実施例では、文字列はマーシャルバッファによってデコードするのに適したオクテットの二進ストリング、つまりストリームである。ASCII文字列は、通常、文字列の初めにヘッダーを含むことに注意しなければならない。ASCII文字列がオクテットの二進ストリームに変換されると、ヘッダーは無視、すなわち完全にマスクされる。オクテットの二進ストリームが形成された後、プロセス制御はステップ352に進み、そこで、オクテットの二進ストリームのためのマーシャルバッファが作成される。次にステップ354で、アンマーシャルメソッドのCORBAオブジェクト版がコールされる。すなわち、図7に関して上述したアンマーシャルメソッドが、インタフェースY用のオブジェクトレファレンスをアンマーシャルするためにコールされる。この場合、既知のタイプのインタフェース、すなわちインタフェースYは、CORBAオブジェクト、すなわちY＝CORBA::Objectである。アンマーシャルメソッドがコールされた後、デストリンジファイされたオブジェクトレファレンスが返される。

【0063】上記に説明したメソッド、すなわちナロー、アンマーシャル、およびデストリンジファイメソッドも、オブジェクトレファレンスを作成するために使用できる。これらのメソッドにはクライアント表示のデュプリケートメソッドをコールできるものもあれば、できないものもある。一般に、ORBに固有のインプリメンテーション定義、インタフェース定義、およびユーザーによって提供されるレファレンスデータが分かっているか与えられると、オブジェクトレファレンスをサーバントから作成することができる。インプリメンテーション定義は現行プロセスのホストIDとサーバーIDとを含む。インプリメンテーション定義、インタフェース定義、およびレファレンスデータが分かっていれば、オブジェクトレファレンスを作成するために明白なオブジェクトアダプタを必要としない。しかしながら、レファレ

ンスデータを明白に規定する必要がある場合は、オブジェクトレファレンスを作成するためにオブジェクトアダプタが必要である。

【0064】次に図12を参照して、本発明の一実施例による、サーバントがローカルかリモートかを決定するオブジェクトレファレンスを作成するメソッドを、オブジェクトアダプタ（OA）を使用してオブジェクトレファレンスの作成に関連して説明する。OAは引数としてインプリメンテーション定義（例えばNEO分散オブジェクトシステムにおけるIMPL_DEF）、インタフェース定義（例えばINTF_DEF）、およびレファレンスデータ（例えばREF_DATA）を使用して、ローカルプロセスのオブジェクトレファレンスインタフェースを作成する。この実施例では、INTF_DEFの使用はオプションである。プロセスはステップ402で開始され、そこで、引数IMPL_DEFを使って、作成すべきオブジェクトレファレンスに関連するサブコントラクトが決定される。IMPL_DEFは、どのサブコントラクトが適切であるかを決定することによって、使用すべきサブコントラクトを識別する。ステップ404では、IMPL_DEFによって識別されたサブコントラクトに適したクライアント表示が、「クリエート」引数、すなわちクライアント表示の作成に必要な情報を規定する引数で提供された情報を使って作成される。適切なクライアント表示が作成された後、プロセスの流れはステップ406に進み、そこで、作成されたクライアント表示の中のホストIDとサーバーIDが、現行プロセスのホストIDおよびサーバーIDと同一か否かの決定が行なわれる。一旦、決定が行なわれると、それに従ってクライアント表示のローカルフラグがセットされる。プロセス制御は、次に、ステップ408に進み、作成されたクライアント表示が現行プロセスに対してローカルか否かを示すためにローカルフラグをセットするか否かの決定が行なわれる。すなわち、ステップ408では、ローカルフラグが正しくセットされているか否かを決定する。決定が肯定的な場合、すなわち、ローカルフラグが正しくセットされている旨決定された場合、ステップ410で、クライアント表示のデュプリケートメソッドへのコールがCORBAオブジェクトローカルmテーブルへのポインタを使って行なわれる。次いで、結果が返される。

【0065】クライアント表示が2つ以上の関連ローカルmテーブルを持つ場合、一旦、ローカルフラグが正しくセットされると、プロセスの流れはステップ410aに移る。ステップ410aは、クライアント表示が関連する2つ以上のローカルmテーブルが存在するときは、ナロー化オブジェクトレファレンスを作成する場合に使用すべき適切なローカルmテーブルの識別を含む。全体ステップ410aは、サーバントオブジェクトの実装のための適切なmテーブルを識別するステップであるステップ411を含む。適切なmテーブルの識別は、サブコントラクト、すなわちクライアント表示固有のステップである。多数の関連mテーブルを持つクライアント表示の場合には、ローカルmテーブルがサーバントオブジェクトの実装用の適切なローカルmテーブルであると判明しない限り、ナローメソッドに対する引数としてパスされるローカルmテーブルは、通常、使用されない。一旦、適切なローカルmテーブルが識別されると、識別されたCORBAオブジェクトローカルmテーブルへのポインタを使って、ステップ413で、クライアント表示のデュプリケートメソッドに対するコールが行なわれる。デュプリケートメソッドへのコールの後、コールの結果が返される。

【0066】ローカルフラグがセットされていないか、間違ってセットされている旨ステップ408で決定された場合、プロセス制御はステップ412に進み、そこで、クライアント表示デュプリケートメソッドに対するコールが、CORBAオブジェクトのリモートmテーブルへのポインタを使って行なわれる。CORBAオブジェクトリモートmテーブルによるクライアント表示のデュプリケートメソッドに対するコールが完了すると、コールの結果が返される。

【0067】先に記載したように、インプリメンテーション定義、インタフェース定義、およびレファレンスデータが提供されると、オブジェクトレファレンスのサーバントベースの作成が、通常、可能である。レファレンスデータが提供されず、それを規定しなければならない場合、図12に関して説明したように、オブジェクトレファレンスを作成するためにオブジェクトアダプタが必要である。しかしながら、レファレンスデータが提供された場合、オブジェクトレファレンスを作成するためのナロー、アンマーシャル、またはデストリンジファイメソッドの使用に代わるものとして、オブジェクトレファレンスを作成するためのサーバー表示の使用がある。サーバー表示は、サーバントと同一のプロセスにあるが、オブジェクトレファレンスを作成するために使用できる。サーバー表示がオブジェクトレファレンスを作成するので、作成されたオブジェクトレファレンスはサーバントと同一プロセスにある。

【0068】次に図13を参照して、サーバー表示を使ってオブジェクトレファレンスを作成するプロセスを説明する。用語のサーバー表示、つまりサーバー「レプ」と、サーバント表示、つまりサーバント「レプ」とは、互換性をもって使用できることを理解しなければならない。リモートとローカルmテーブルを引数として、サーバントに付されたサーバー表示によってオブジェクトレファレンスが作成される。ステップ450では、サーバー表示に対応するクライアント表示が作成される。関連するサブコントラクトに適したクライアント表示の作成に必要な引数は、サーバー表示の中で発見することがで

きる。クライアント表示が作成された後、プロセス制御はステップ452に進み、そこで、クライアント表示とローカルmテーブルへのポインタとを含むオブジェクト、つまりオブジェクトレファレンスが作成される。次に、ステップ454で、新たに作成されたオブジェクトレファレンスが返される前にサーバーと同一プロセスのクライアント表示が作成されたことを示すために、ローカルフラグを正しくセットすることができる。ローカルフラグを正しくセットすることは、クライアント表示、すなわちオブジェクトレファレンスが、サーバントと同一プロセスにあることを示す。

【0069】本発明は、上記のように、コンピュータシステムに格納されるデータを伴う様々なプロセスステップを使用する。これらのステップは物理量の物理的操作を必要とするステップである。通常、必ずしもそうではないが、これらの量は格納、転送、結合、比較、その他の操作が可能な電気的または磁気的信号の形を取る。主として一般的慣行の理由で、これらの信号をビット、バリュー、エレメント、変数、キャラクタ、データ構造等と呼ぶと便利な場合がある。しかしながら、これらの用語や類似の用語は、すべて、適切な物理量に関連されるべきであり、また、これらの量に付される単なる便利なラベルに過ぎないことを忘れてはならない。

【0070】更に、実行される操作は識別、実行、または比較等の用語で呼ばれることが多い。本発明の一部を形成する、本明細書に記載のオペレーションでは、これらのオペレーションはマシンオペレーションである。本発明のオペレーションを実行するために便利なマシンは、汎用ディジタルコンピュータやその他類似の装置を含む。いずれにしても、コンピュータを操作する場合のオペレーションのメソッドと、計算自体のメソッドとの相違を忘れてはならない。本発明は、電気的またはその他の物理的信号を処理して他の望ましい物理信号を発生させる場合の、コンピュータを操作するためのメソッドステップに関する。

【0071】本発明は、これらのオペレーションを実行する装置にも関係する。この装置は、要求される目的のために特別に製造してもよいし、また、それはコンピュータに格納されたコンピュータプログラムによって選択的に作動されるか再構成される汎用コンピュータでもよい。本明細書で提示されるプロセスは、特定のコンピュータやその他装置に固有の関係を持つものではない。特に本明細書の教示に従って書かれたプログラムを使って様々な汎用マシンを使用できるし、また、要求されるメソッドステップを実行するためにより特殊化された装置を製造する方が便利な場合もある。これらの様々なマシンに必要な構造は上記の説明から明らかであろう。

【0072】また、本発明は、更に、様々なコンピュータ実装オペレーションを実行するためのプログラム命令を含むコンピュータ読取り可能メディアに関係する。メディアとプログラム命令は、本発明の目的のために特別に設計、製造されたものでよいし、また、コンピュータソフトウェア技術の専門家に周知の利用可能な類のものでもよい。コンピュータ読取り可能メディアの例は、これに限定されるわけではないが、ハードディスク、フロッピーディスク、および磁気テープ等の磁気メディア；CD－ROMディスク等の光学メディア；オプティカルディスク等の磁気光学的メディア；およびプログラム命令を格納、実行するために特別に構成されたハードウェアディバイス、例えばリードオンリーメモリ（ROM）およびランダムアクセスメモリ（RAM）を含む。プログラム命令の例は、コンパイラによって製作されたマシンコードと、インタプリタを使ってコンピュータによって実行されるハイレベルのコードを含むファイルの両者を含む。

【0073】図14は、本発明による典型的なコンピュータシステムを示す。コンピュータシステム500は、主記憶装置504（通常、リードオンリーメモリ、またはROMと呼ぶ）、および主記憶装置506（通常、ランダムアクセスメモリ、またはRAMと呼ぶ）を含む記憶装置に連結された任意の数のプロセッサ502（中央処理装置、つまりCPUとも呼ぶ）を含む。当業者には周知のように、ROM504は、データと命令をCPUに一方向的に転送する働きがあり、RAM506は、通常、データと命令を双方向的に転送するために使用される。主記憶装置504、506は、共に、上記のように任意の適切なコンピュータ読取り可能媒体を含むことができる。大容量記憶装置508もCPU502に双方向的に連結されて、追加のデータ記憶容量を提供する。大容量記憶装置508はプログラム、データ等を格納するために使用され、通常、主記憶装置504、506よりも遅いハードディスク等の二次記憶媒体である。大容量記憶装置508は、磁気または紙テープリーダーやその他の周知の装置の形を取ることができる。大容量記憶装置508内に保持される情報は、適切な場合に、バーチュアルメモリとして、RAM506の一部として標準的に組み込まれることはよく理解されるだろう。CD－ROM514等の特定の大容量記憶装置もデータを一方向的にCPUに送る。

【0074】CPU502も、これに限定されるわけではないが、ビデオモニタ、トラックボール、マウス、キーボード、マイクロホン、タッチセンシティブディスプレイ、トランスデューサカードリーダ、磁気または紙テープレコーダ、タブレット、スタイラス、音声や手書きの認識装置、その他の周知の入力装置、例えば、勿論、他のコンピュータ等の、一つ以上の入出力装置510に連結される。最後に、CPU502は、オプションとして、512で概略的に示されるネットワーク接続を使用してコンピュータや通信ネットワークに連結することができる。上記のネットワーク接続によって、CPUは、

上記のメソッドステップを実行する中でネットワークからの情報を受け取ったり、ネットワークに情報を出力するものと予想される。上記の装置と材料は、コンピュータハードウェアとソフトウェア技術の専門家には周知のものであろう。

【００７５】本発明の実施例を一つだけ説明したが、本発明は、その精神と範囲から逸脱することなく、他の多くの形式で具体化し得ることは言うまでもない。特に、本発明に関して記載されたmテーブルは多くの異なる形を取ることができる。例として、これらの形は、これに限定されるわけではないが、従来のC++vテーブルに似た構造を持つフラットmテーブル、およびツリー構造のmテーブルを含む。従って、記載の実施例は説明のためであって、限定のためではないと見做すべきであり、本発明は下記クレームならびにそれらに相当する仕様によって定義されるものとする。

【図面の簡単な説明】

【図１】分散オブジェクトシステムの象徴的な概要図である。

【図２】クライアントによるリクエストが、分散オブジェクトシステムのクライアント側とサーバー側のアーキテクチャ、および分散オブジェクトシステムのクライアント側とサーバー側の間のインターフェースを介してどのように流れるかを示す概略説明図である。

【図３】オブジェクトレファレンスの概略表示図である。

【図４】ファットポインタ、クライアント表示、および分散オブジェクトレファレンスに関連するメソッドの間のインタフェースの概略説明図である。

【図５】本発明の一実施例による、オブジェクトの呼出に係わるステップを説明するプロセス流れ図である。

【図６】本発明の一実施例による、ローカルスタブの処理に係わるステップを説明するプロセス流れ図である。

【図７】本発明の一実施例による、オブジェクトレファレンスを複製する(duplicating) 処理に係わるステップを説明するプロセス流れ図である。

【図８】本発明の一実施例による、オブジェクトレファレンスをナロー化する(narrowing) 処理に係わるステップを説明するプロセス流れ図である。

【図９】本発明の一実施例による、オブジェクトレファレンスをアンマーシャル(unmarshaling)する処理に係わるステップを説明するプロセス流れ図である。

【図１０】本発明の一実施例による、図７に示すオブジェクトレファレンスをアンマーシャルする処理においてマーシャルバッファの情報からクライアント表示を作成するステップのための代替メソッドを説明するプロセス流れ図である。

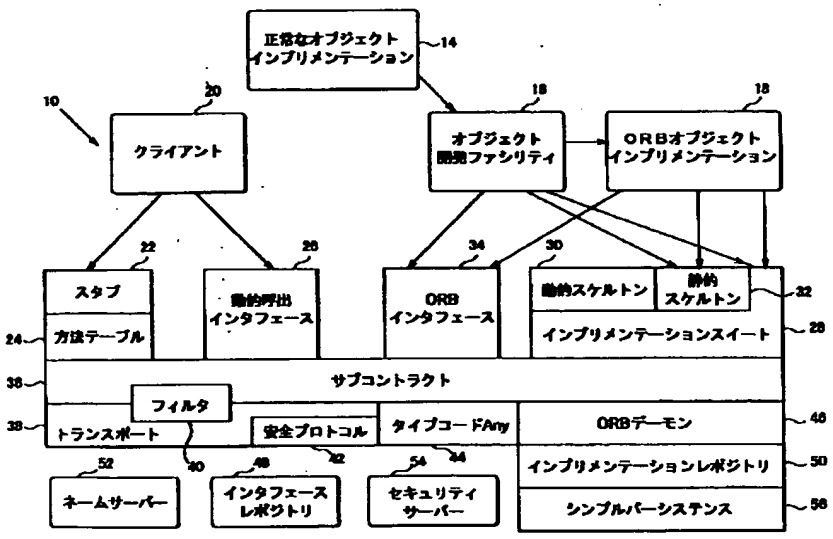
【図１１】本発明の一実施例による、オブジェクトレファレンスをデストリンジファイする(destringifying)処理に係わるステップを説明するプロセス流れ図である。

【図１２】本発明の一実施例による、オブジェクトレファレンスを作成するためのオブジェクトアダプタの使用に係わるステップを説明するプロセス流れ図である。

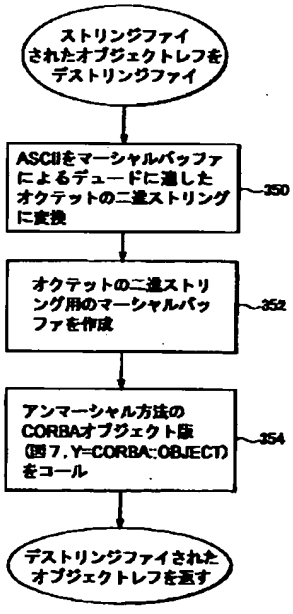
【図１３】本発明の一実施例による、オブジェクトレファレンスを作成するためのサーバー表示の使用に係わるステップを説明するプロセス流れ図である。

【図１４】本発明による代表的なコンピュータシステムの説明図である。

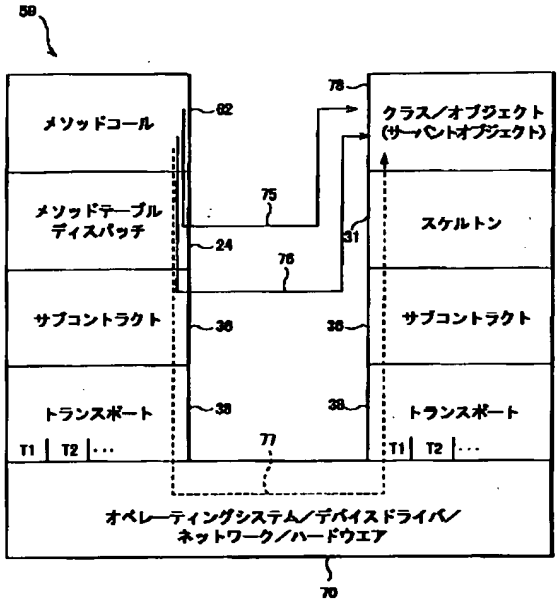
【図1】



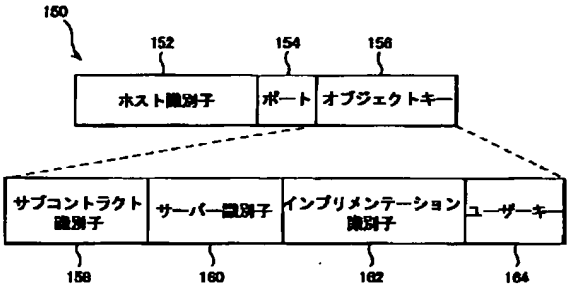
【図11】



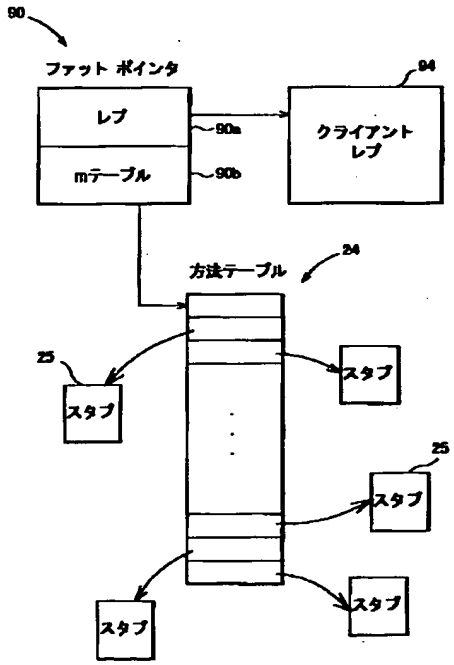
【図2】


59
メソッドコール
62
78
クラス／オブジェクト
(サーバントオブジェクト)
メソッドテーブル
ディスパッチ
75
24
76
31
スケルトン
サブコントラクト
36
36
サブコントラクト
トランスポート
38
38
トランスポート
T1
T2
…
77
T1
T2
…
オペレーティングシステム／デバイスドライバ／
ネットワーク／ハードウエア
70

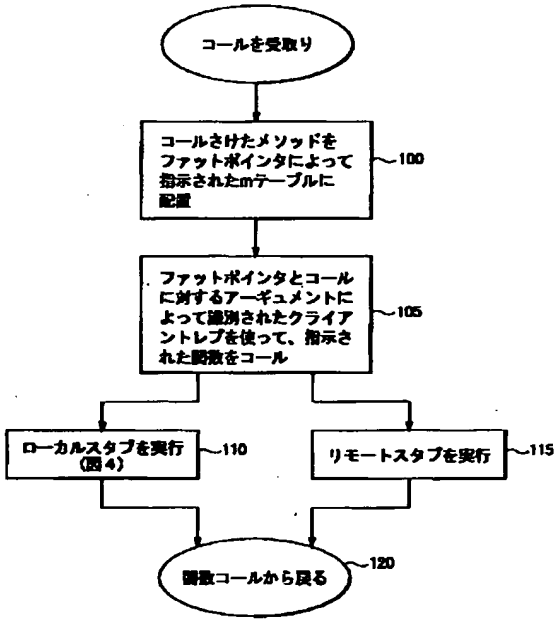
【図3】


150
152
154
156
ホスト識別子
ポート
オブジェクトキー
サブコントラクト
識別子
サーバー識別子
インプリメンテーション
識別子
ユーザーキー
158
160
162
164

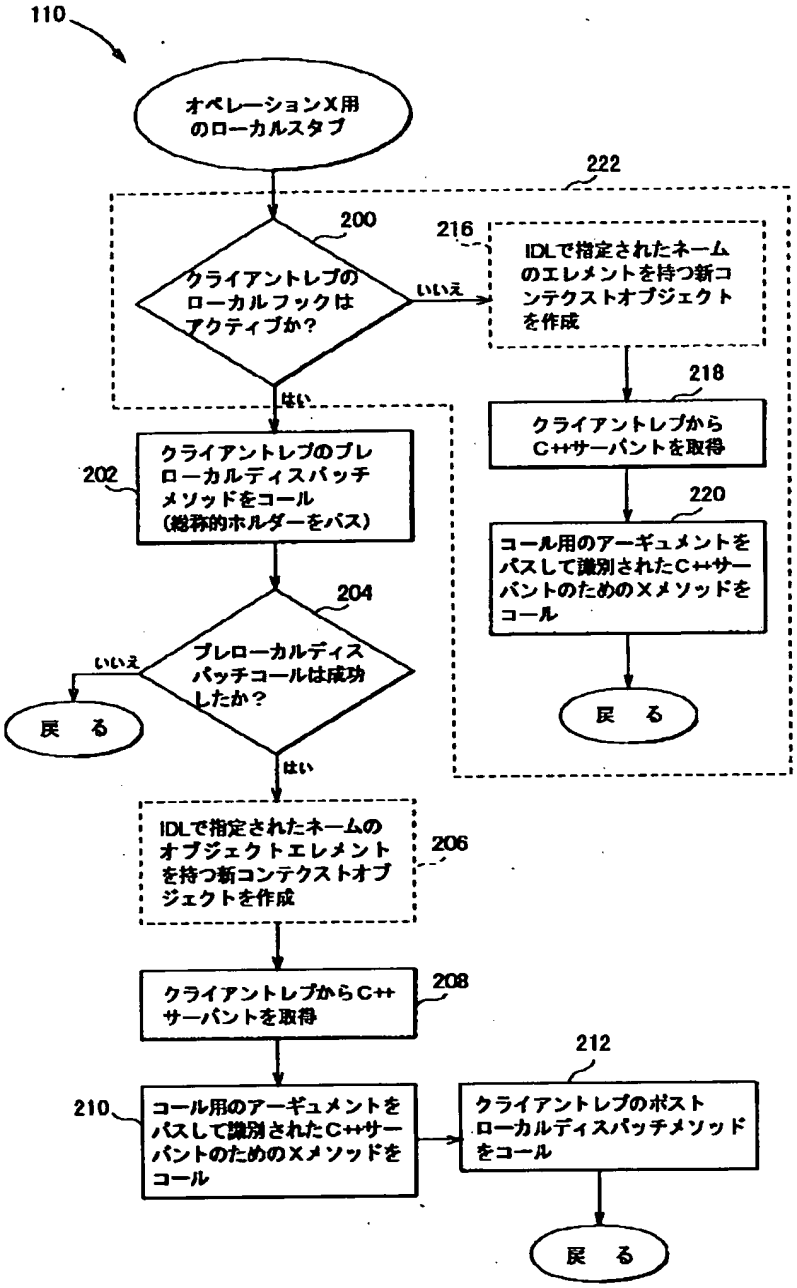
【図4】


90
ファット ポインタ
レプ
90a
mテーブル
90b
94
クライアント
レプ
方法テーブル
24
25
スタブ
スタブ
25
スタブ
スタブ
スタブ

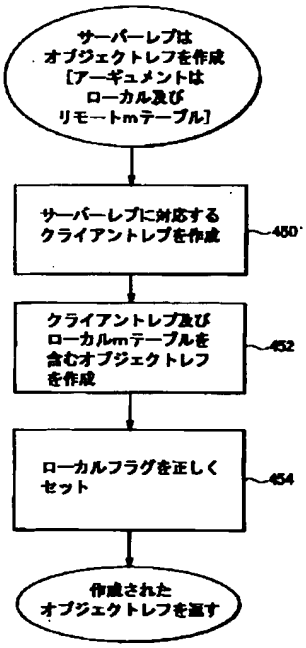
【図5】


コールを受取り
コールさけたメソッドを
ファットポインタによって
指示されたmテーブルに
配置
100
ファットポインタとコール
に対するアーギュメントに
よって識別されたクライア
ントレプを使って、指示さ
れた関数をコール
105
ローカルスタブを実行
(図4)
110
リモートスタブを実行
115
関数コールから戻る
120

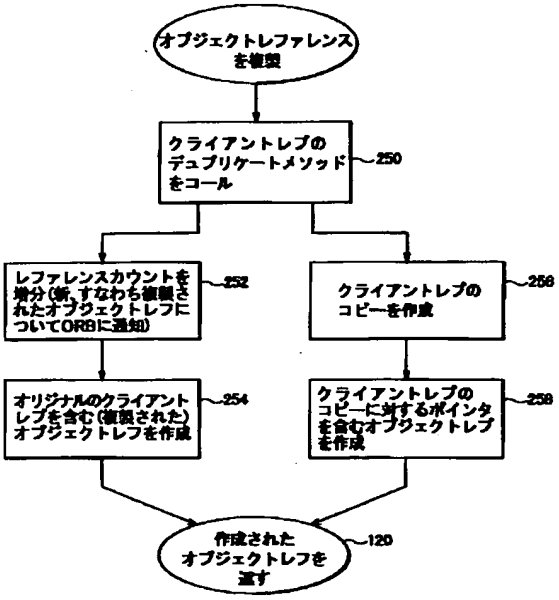
【図6】


110
オペレーションX用のローカルスタブ
222
200
クライアントレプのローカルフックはアクティブか?
いいえ
216
IDLで指定されたネームのエレメントを持つ新コンテクストオブジェクトを作成
218
クライアントレプからC++サーバントを取得
220
コール用のアーギュメントをパスして識別されたC++サーバントのためのXメソッドをコール
戻る
はい
202
クライアントレプのプレローカルディスパッチメソッドをコール(総称的ホルダーをパス)
204
プレローカルディスパッチコールは成功したか?
いいえ
戻る
はい
IDLで指定されたネームのオブジェクトエレメントを持つ新コンテクストオブジェクトを作成
206
クライアントレプからC++サーバントを取得
208
210
コール用のアーギュメントをパスして識別されたC++サーバントのためのXメソッドをコール
212
クライアントレプのポストローカルディスパッチメソッドをコール
戻る

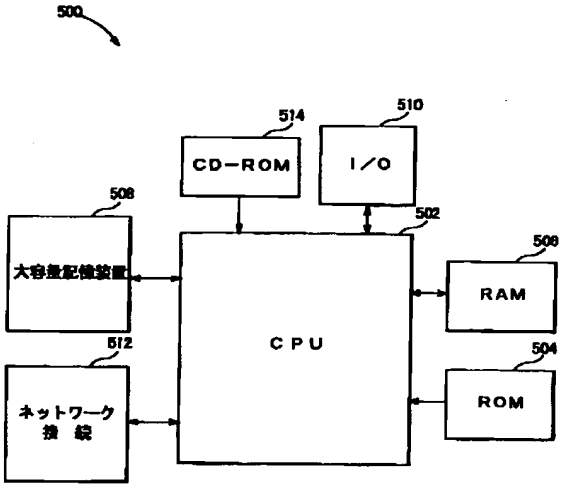
【図13】


サーバーレプはオブジェクトレフを作成[アーギュメントはローカル及びリモートmテーブル]
サーバーレプに対応するクライアントレプを作成
450
クライアントレプ及びローカルmテーブルを含むオブジェクトレフを作成
452
ローカルフラグを正しくセット
454
作成されたオブジェクトレフを返す

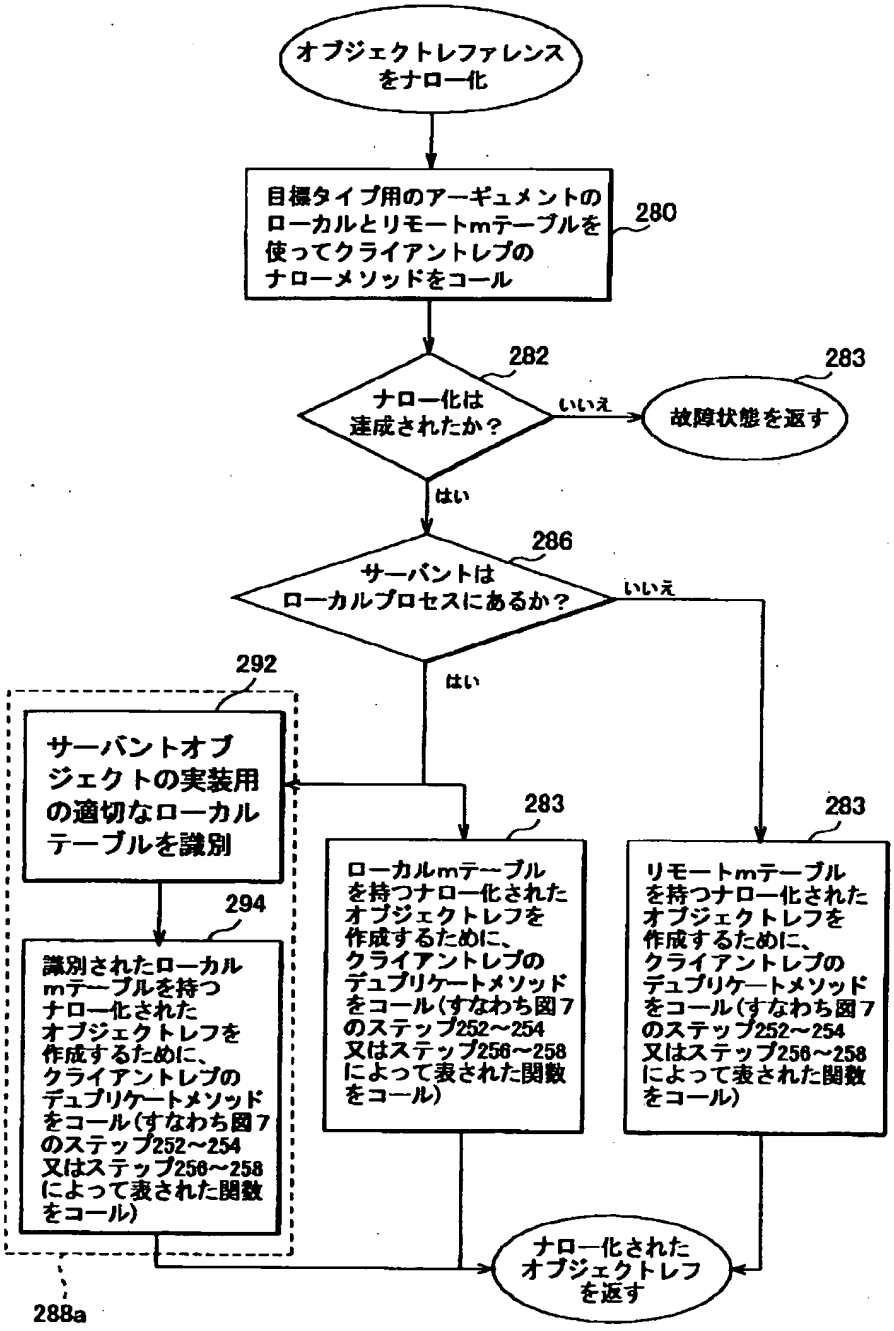
【図7】


オブジェクトレファレンスを複製
クライアントレプのデュプリケートメソッドをコール
250
レファレンスカウントを増分(新、すなわち複製されたオブジェクトレフについてORBに通知)
252
クライアントレプのコピーを作成
256
オリジナルのクライアントレプを含む(複製された)オブジェクトレフを作成
254
クライアントレプのコピーに対するポインタを含むオブジェクトレプを作成
258
作成されたオブジェクトレフを返す
120

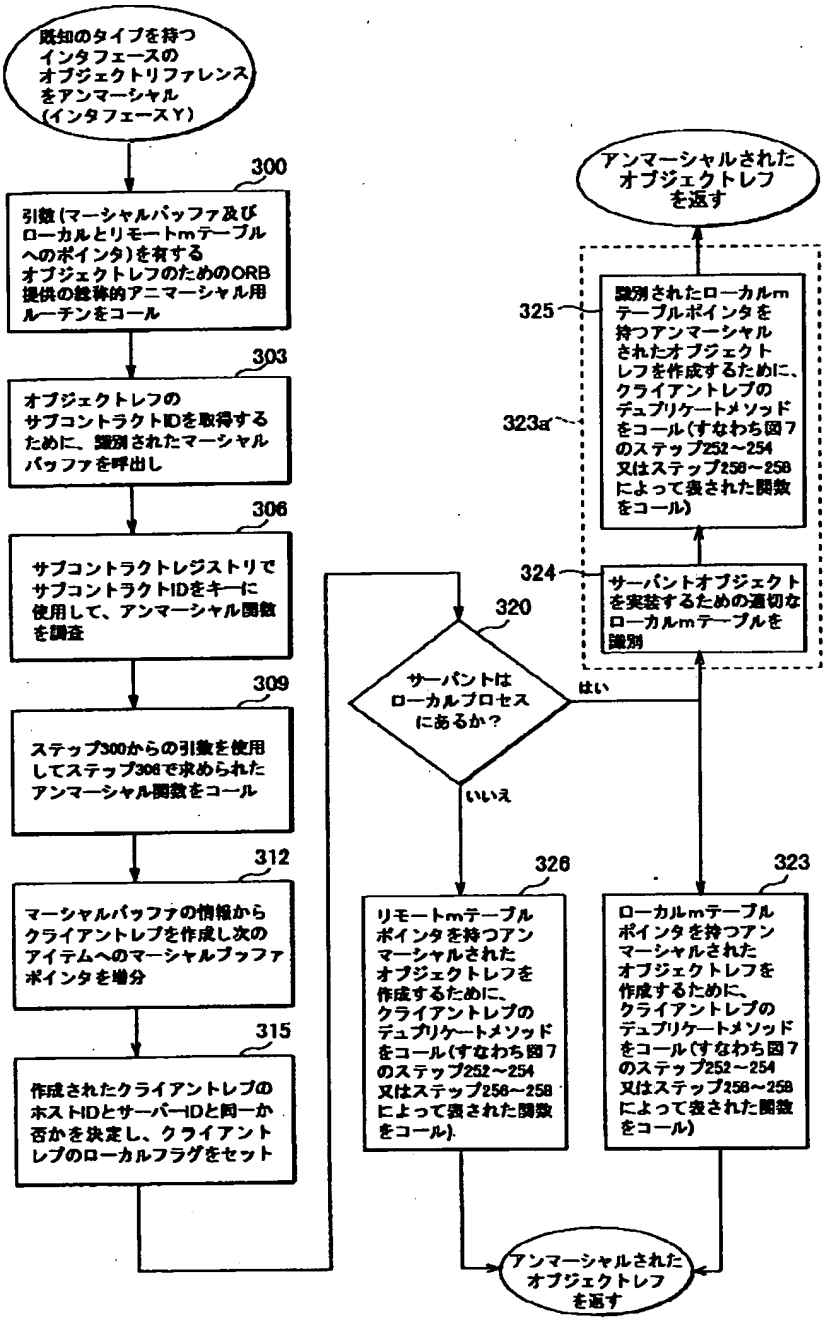
【図14】


500
514
CD-ROM
510
I/O
508
大容量記憶装置
502
506
RAM
CPU
512
ネットワーク接続
504
ROM

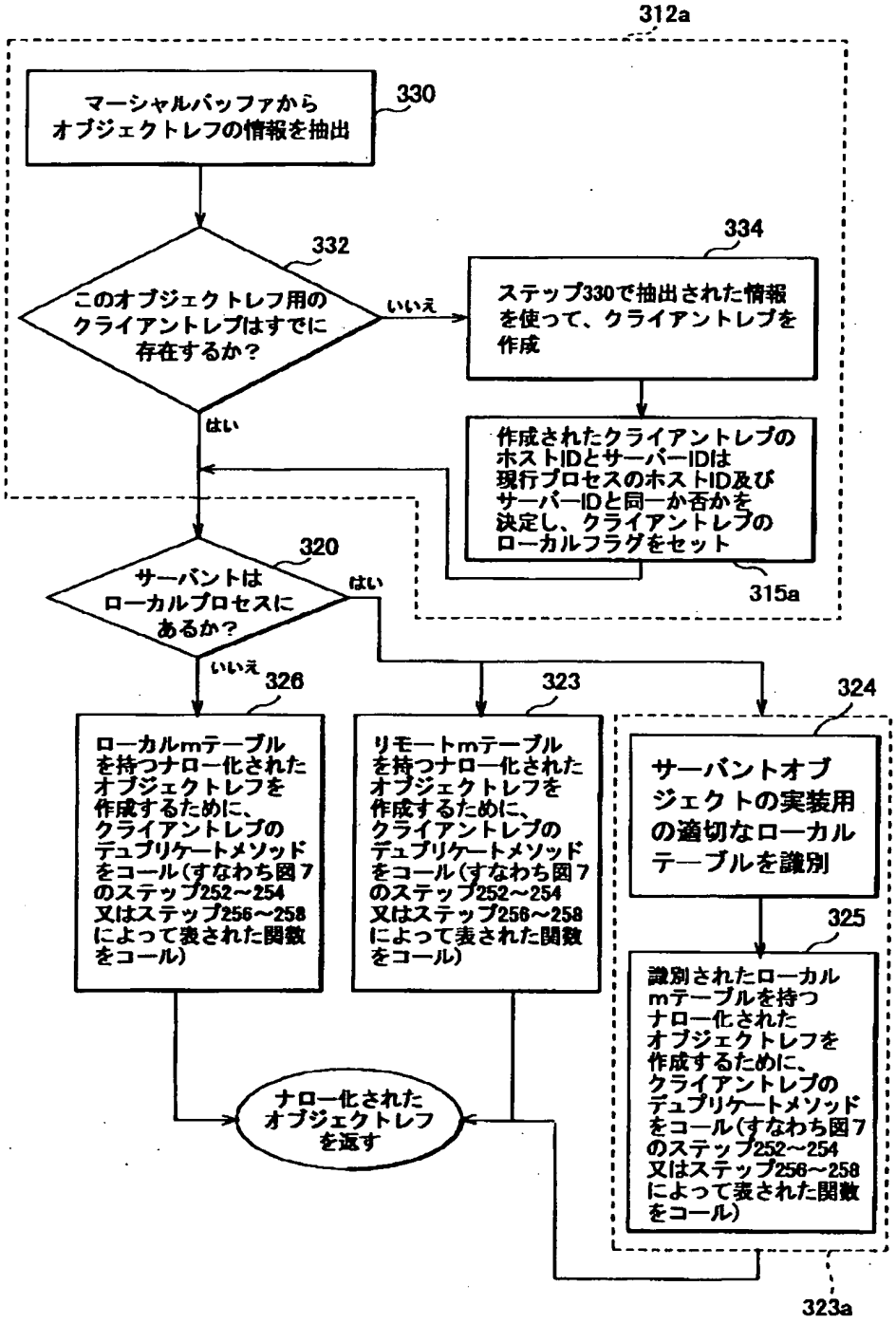
【図8】


オブジェクトレファレンス
をナロー化
目標タイプ用のアーギュメントの
ローカルとリモートmテーブルを
使ってクライアントレプの
ナローメソッドをコール
280
282
ナロー化は
達成されたか？
いいえ
283
故障状態を返す
はい
286
サーバントは
ローカルプロセスにあるか？
いいえ
はい
292
サーバントオブ
ジェクトの実装用
の適切なローカル
テーブルを識別
294
識別されたローカル
mテーブルを持つ
ナロー化された
オブジェクトレフを
作成するために、
クライアントレプの
デュプリケートメソッド
をコール(すなわち図7
のステップ252～254
又はステップ256～258
によって表された関数
をコール)
283
ローカルmテーブル
を持つナロー化された
オブジェクトレフを
作成するために、
クライアントレプの
デュプリケートメソッド
をコール(すなわち図7
のステップ252～254
又はステップ256～258
によって表された関数
をコール)
283
リモートmテーブル
を持つナロー化された
オブジェクトレフを
作成するために、
クライアントレプの
デュプリケートメソッド
をコール(すなわち図7
のステップ252～254
又はステップ256～258
によって表された関数
をコール)
ナロー化された
オブジェクトレフ
を返す
288a

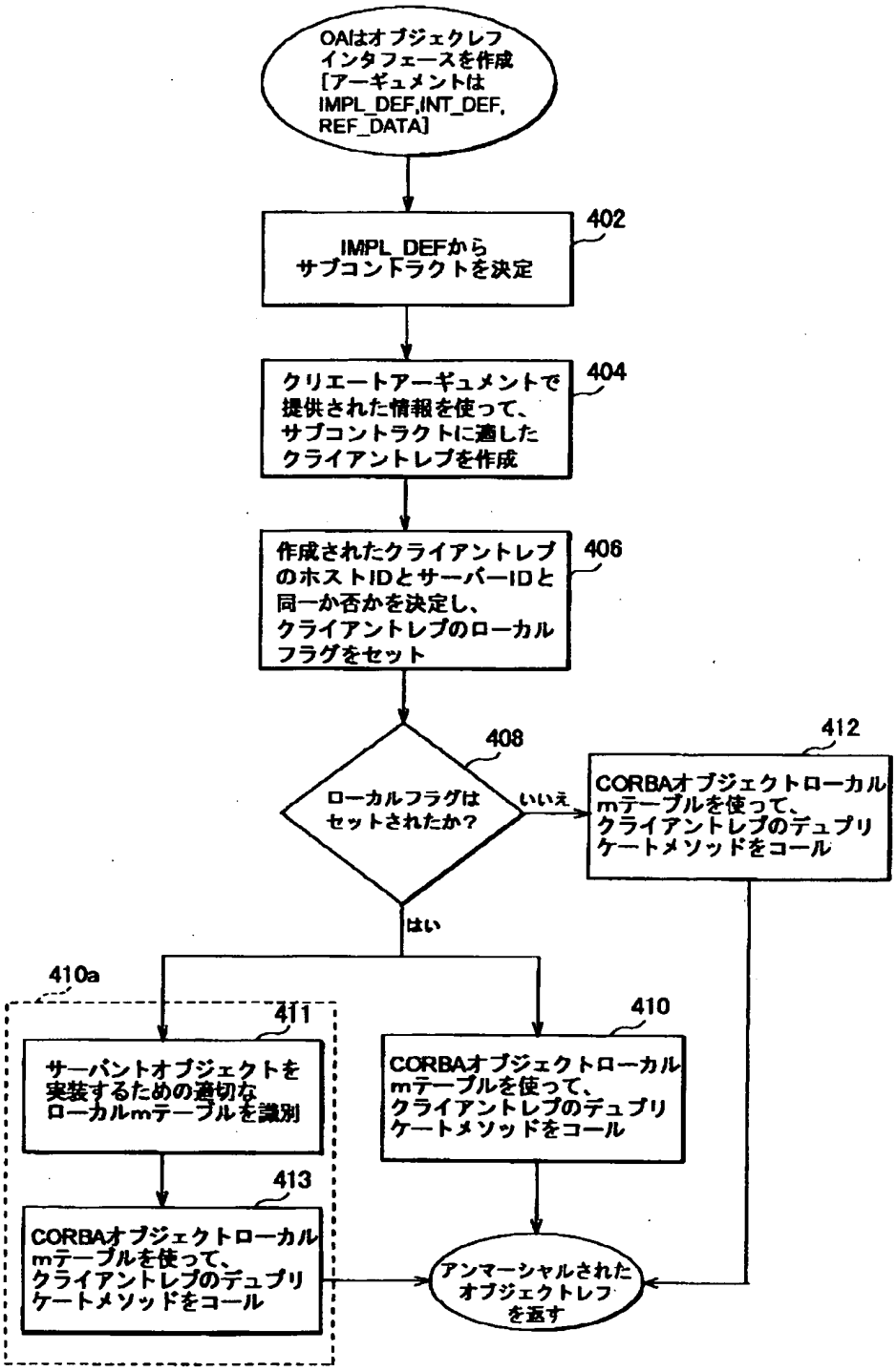
【図9】


既知のタイプを持つインタフェースのオブジェクトリファレンスをアンマーシャル(インタフェースY)
300
引数(マーシャルバッファ及びローカルとリモートmテーブルへのポインタ)を有するオブジェクトレフのためのORB提供の総称的アンマーシャル用ルーチンをコール
303
オブジェクトレフのサブコントラクトIDを取得するために、識別されたマーシャルバッファを呼出し
306
サブコントラクトレジストリでサブコントラクトIDをキーに使用して、アンマーシャル関数を調査
309
ステップ300からの引数を使用してステップ306で求められたアンマーシャル関数をコール
312
マーシャルバッファの情報からクライアントレプを作成し次のアイテムへのマーシャルバッファポインタを増分
315
作成されたクライアントレプのホストIDとサーバーIDと同一か否かを決定し、クライアントレプのローカルフラグをセット
320
サーバントはローカルプロセスにあるか?
はい
いいえ
アンマーシャルされたオブジェクトレフを返す
325
識別されたローカルmテーブルポインタを持つアンマーシャルされたオブジェクトレフを作成するために、クライアントレプのデュプリケートメソッドをコール(すなわち図7のステップ252～254又はステップ256～258によって表された関数をコール)
323a'
324
サーバントオブジェクトを実装するための適切なローカルmテーブルを識別
326
リモートmテーブルポインタを持つアンマーシャルされたオブジェクトレフを作成するために、クライアントレプのデュプリケートメソッドをコール(すなわち図7のステップ252～254又はステップ256～258によって表された関数をコール)
323
ローカルmテーブルポインタを持つアンマーシャルされたオブジェクトレフを作成するために、クライアントレプのデュプリケートメソッドをコール(すなわち図7のステップ252～254又はステップ256～258によって表された関数をコール)
アンマーシャルされたオブジェクトレフを返す

【図10】


312a
マーシャルバッファから
オブジェクトレフの情報を抽出
330
このオブジェクトレフ用の
クライアントレプはすでに
存在するか？
332
いいえ
ステップ330で抽出された情報
を使って、クライアントレプを
作成
334
はい
作成されたクライアントレプの
ホストIDとサーバーIDは
現行プロセスのホストID及び
サーバーIDと同一か否かを
決定し、クライアントレプの
ローカルフラグをセット
315a
サーバントは
ローカルプロセスに
あるか？
320
はい
いいえ
ローカルmテーブル
を持つナロー化された
オブジェクトレフを
作成するために、
クライアントレプの
デュプリケートメソッド
をコール（すなわち図7
のステップ252～254
又はステップ256～258
によって表された関数
をコール）
326
リモートmテーブル
を持つナロー化された
オブジェクトレフを
作成するために、
クライアントレプの
デュプリケートメソッド
をコール（すなわち図7
のステップ252～254
又はステップ256～258
によって表された関数
をコール）
323
サーバントオブ
ジェクトの実装用
の適切なローカル
テーブルを識別
324
識別されたローカル
mテーブルを持つ
ナロー化された
オブジェクトレフを
作成するために、
クライアントレプの
デュプリケートメソッド
をコール（すなわち図7
のステップ252～254
又はステップ256～258
によって表された関数
をコール）
325
ナロー化された
オブジェクトレフ
を返す
323a

【図12】


OAIはオブジェクレフ
インタフェースを作成
[アーギュメントは
IMPL_DEF,INT_DEF,
REF_DATA]
IMPL_DEFから
サブコントラクトを決定
402
クリエートアーギュメントで
提供された情報を使って、
サブコントラクトに適した
クライアントレプを作成
404
作成されたクライアントレプ
のホストIDとサーバーIDと
同一か否かを決定し、
クライアントレプのローカル
フラグをセット
406
ローカルフラグは
セットされたか？
408
いいえ
CORBAオブジェクトローカル
mテーブルを使って、
クライアントレプのデュプリ
ケートメソッドをコール
412
はい
410a
サーバントオブジェクトを
実装するための適切な
ローカルmテーブルを識別
411
CORBAオブジェクトローカル
mテーブルを使って、
クライアントレプのデュプリ
ケートメソッドをコール
410
CORBAオブジェクトローカル
mテーブルを使って、
クライアントレプのデュプリ
ケートメソッドをコール
413
アンマーシャルされた
オブジェクトレフ
を返す

フロントページの続き

(72)発明者　ピーター　ビー．　ケスラー
アメリカ合衆国，　カリフォルニア州，
パロ　アルト，　ヴァードサ　ドライヴ
4121

(72)発明者　サンジェイ　アール．　ラディア
アメリカ合衆国，　カリフォルニア州，
フリーモント，　ボアー　サークル　883

(72)発明者　グラハム　ハミルトン
アメリカ合衆国，　カリフォルニア州，
パロ　アルト，　デイヴィッド　コート
3143